(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
10 janvier 2002 (10.01.2002)

PCT

(10) Numéro de publication internationale
WO 02/03245 A1

(51) Classification internationale des brevets⁷ : G06F 17/30

(21) Numéro de la demande internationale :
PCT/FR00/01902

(22) Date de dépôt international : 4 juillet 2000 (04.07.2000)

(25) Langue de dépôt : français

(26) Langue de publication : français

(71) Déposant *(pour tous les États désignés sauf US)* : **OTOOBE** [FR/FR]; 46, rue Lauriston, F-75116 Paris (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)* : **QUEREL, Laurent** [FR/FR]; 40, rue du Docteur Foucault, F-92000 Nanterre (FR). **SAIM HADDACHE, Boubakar** [DZ/FR]; 14, rue de Torcy, F-75018 Paris (DZ). **FONTVIELLE, Laurent** [FR/FR]; 33bis, rue Thomas Lemaître, F-92000 Nanterre (FR).

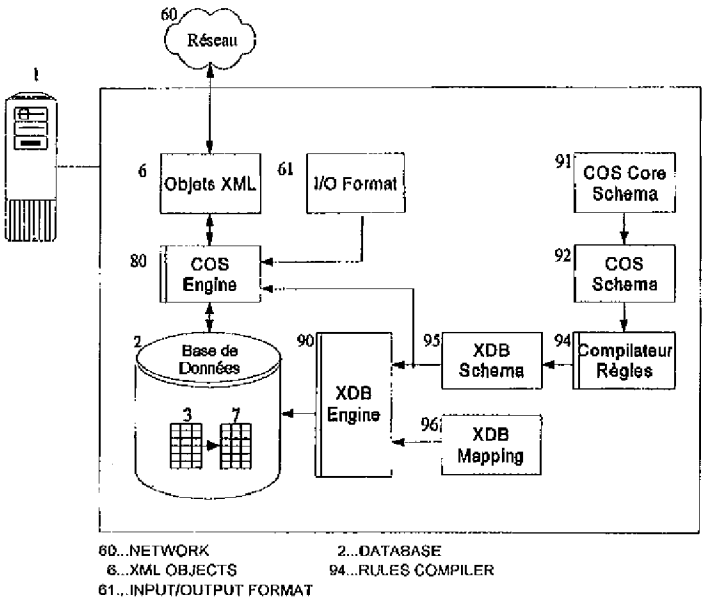
(74) Mandataire : **BOUTIN, Antoine**; Cabinet Loyer, 78, avenue Raymond Poincaré, F-75116 Paris (FR).

(81) États désignés *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE,

*[Suite sur la page suivante]*

(54) Title: METHOD FOR STORING XML-FORMAT INFORMATION OBJECTS IN A RELATIONAL DATABASE

(54) Titre : PROCEDE DE STOCKAGE D'OBJETS INFORMATIONNELS AU FORMAT XML DANS UNE BASE DE DONNEES RELATIONNELLE




**(57) Abstract:** The invention concerns a method for storing information objects or objects (5) in a relational database (2) stored in a server computer (1), the relational database (2) consisting of tables, each formed by a table of single data sets having the same data structure in a common table, each single data being designated by a single identifier in said table, an object (6) consisting of one or several single data capable of being stored in a table of the database (2), and/or one or several objects nested in said object. The nesting can be produced on any number of levels to produce an object (6), a nested object or a single data said to be locally in fixed number if it appears exactly once in the object immediately containing it and said to be locally in variable number otherwise. A single data occurring at a particular level of said object (6) is said to be globally in fixed number if it is locally fixed and all the objects containing it are in fixed number, a single data occurring at a particular level of said object is said to be invariable number if it is locally in variable number or if any one of the objects containing it is locally in variable number. The data globally in fixed number of an object to be stored (6) are stored in a main data set (31) stored in a main table (3) of the database (2). The single data globally in variable number of said object to be stored (6) are stored in one or several auxiliary tables (4, 5, 7) of the database (2). When they exist, the single data globally in variable number of said objects (6) are stored in a single auxiliary table (7) of the database. The method create one or several sets of auxiliary data sets (71) to store single data globally in variable number in the single auxiliary table (7).

**(57) Abrégé :** Procédé de stockage d'objets informationnels ou objets (6), dans une base de données relationnelle (2) stockée dans un ordinateur serveur (1), la base de données relationnelle (2) étant constituée de tables, chacune constituée d'un tableau d'ensembles de données simples qui ont

*[Suite sur la page suivante]*

WO 02/03245 A1

DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

**(84) États désignés** *(régional)* : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

**Publiée :**

— *avec rapport de recherche internationale*

*En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.*

la même structure de données dans une même table, chaque donnée simple d'une table est désignée par un identifiant unique dans ladite table, un objet (6) étant constitué d'une ou plusieurs données simples pouvant être stockées dans une table de la base de données (2), et/ou d'un ou plusieurs objets emboîtés dans ledit objet. L'emboîtement peut être réalisé sur un nombre quelconque de niveaux pour réaliser un objet (6), un objet emboîté ou une donnée simple étant dit localement en nombre fixe s'il ou elle apparaît exactement une fois dans l'objet le ou la contenant immédiatement et étant dit localement en nombre variable dans le cas contraire. Une donnée simple apparaissant à un niveau quelconque dudit objet (6) est dite globalement en nombre fixe si elle est localement en nombre fixe et si tous les objets la contenant sont localement en nombre fixe, une donnée simple apparaissant à un niveau quelconque dudit objet (6) est dite globalement en nombre variable si elle est localement en nombre variable ou si l'un quelconque des objets la contenant est localement en nombre variable. Les données globalement en nombre fixe d'un objet à stocker (6) sont stockées dans un ensemble de données principal (31) stocké dans une table principale (3) de la base de données (2). Les données simples globalement en nombre variable dudit objet à stocker (6) sont stockées dans une ou plusieurs tables annexes (4, 5, 7) de la base de données (2). Lorsqu'elles existent, les données simples globalement en nombre variable desdits objets (6) sont stockées dans une unique table annexe (7) de la base de données, le procédé crée un ou plusieurs ensembles de données annexes (71) pour stocker les données simples globalement en nombre variable dans l'unique table annexe (7).

# PROCEDE DE STOCKAGE D'OBJETS INFORMATIONNELS AU FORMAT XML DANS UNE BASE DE DONNEES RELATIONNELLE

La présente invention a pour objet les procédés de stockage d'objets informationnels, ou objets, dans un système informatique, et plus particulièrement, un procédé de stockage, dans une base de données relationnelle traditionnelle, d'objets au format eXtended Markup Language, ou XML, du consortium Internet World Wide Web, ou W3C.

Les systèmes de stockage de données sont bien connus dans les techniques de traitement de l'information, et de plus en plus fréquemment, ils mettent en œuvre des systèmes de gestion de bases de données relationnelles.

Ces systèmes sont bien adaptés au stockage de données comportant des parties en nombre fixe et des parties en nombre variable, les parties en nombre fixe étant stockées dans une table principale et les parties en nombre variable étant chacune stockées dans une table annexe de la base de données. En général, ces systèmes permettent en outre de cascader les données en nombre variable, c'est-à-dire que chaque ensemble ou enregistrement de données en nombre variable peut être à son tour associé à d'autres données en nombre variable. En outre, ces systèmes de gestion de base de données offrent habituellement des possibilités d'interrogation sophistiquée, et de bonnes performances, même sous forte charge.

En l'absence de systèmes efficaces de gestion de bases de données orientés objets, il est tentant d'utiliser les systèmes de gestion de bases de données relationnelles du commerce pour la gestion et le stockage ou la gestion d'objets. Toutefois, ces systèmes sont mal adaptés à la gestion d'objets. En effet, dans ces systèmes traditionnels, les données sont stockées dans des tables d'ensembles de données ayant une structure fixe, ces ensembles de données étant constitués de données simples. Cela signifie en particulier qu'il n'est pas possible d'imbriquer ou d'emboîter des ensembles de données à l'intérieur d'un autre, même si les ensembles de données sont en nombre fixe.

Ce dernier point pose problème pour la gestion d'objets en ce que l'emboîtement d'objets les uns à l'intérieur des autres est le fondement même de la gestion de données orientée objets. Il est théoriquement possible de résoudre ce problème automatiquement en mettant "à plat" la structure de l'objet, c'est-à-dire, en considérant que tous les éléments de sous-objets inclus dans un objet principal sont en fait des éléments de l'objet principal.

Cela introduit un nouveau problème en ce que les noms d'éléments contenus dans des sous-objets distincts d'un même objet peuvent parfaitement être identiques

et que les noms d'éléments dans un même objet doivent nécessairement être distincts, pour d'évidentes raisons de non-ambiguïté. Ce nouveau problème peut à son tour être résolu automatiquement en incluant le nom du sous-objet contenant dans le nouveau nom "à plat" d'un élément d'un sous-objet. En pratique, cela revient à ajouter les noms de tous les objets contenants à l'élément contenu pour obtenir un nom "à plat" non ambigu.

Malheureusement, cette technique conduit rapidement à des noms "à plat" ayant une longueur de nom dépassant les limites autorisées par les systèmes de gestion de bases de données, même pour des éléments peu imbriqués, du fait que ces systèmes imposent des limites de l'ordre de quelques dizaines de caractères sur la longueur des noms des données simples stockées dans une table. Cette nouvelle difficulté est habituellement résolue en tronquant le nom "à plat" obtenu à la longueur permise par le système. Toutefois, ce procédé conduit usuellement à des noms "à plat" dupliqués.

La résolution définitive de ce problème implique habituellement une intervention manuelle d'un opérateur humain, qui définit lui-même les noms à utiliser dans le système de gestion de base de données, pour chaque élément d'objet imbriqué. De façon évidente, cette intervention manuelle est sujette à erreur et elle est coûteuse en temps humain.

Par ailleurs, la technique consistant à utiliser une table annexe pour chaque partie en nombre variable d'un objet conduit rapidement à des performances déplorables lorsqu'un objet comporte de nombreuses parties en nombre variable, comme c'est souvent le cas en pratique. En effet, chaque accès à un objet stocké dans la base de données requérra un accès à la base de données pour la table principale plus autant d'accès à la base de données qu'il y a de tables annexes, c'est-à-dire, de parties d'objets en nombre variable dans un objet. Si l'objet comporte, comme il est fréquent, cinq ou dix parties en nombre variable, le nombre d'accès à la base de données sera multiplié par cinq ou dix par rapport à un objet n'ayant pas de parties variables, ce se traduit par une dégradation des performances dans un même rapport.

Cette dégradation peut devenir critique dans les systèmes de stockage de données très sollicités, tels que les serveurs Internet, car il n'est pas toujours possible de dupliquer les serveurs pour répartir la charge, en particulier lorsque les données présentes sur un serveur peuvent être modifiées. En outre, cela augmente dans les mêmes proportions la fragilité du système de stockage vis-à-vis d'une panne de courant ou de tout autre problème système entraînant un arrêt inopiné du système informatique hébergeant la base de données.

Il existe donc un besoin pour un procédé qui permette d'adapter les systèmes de gestion de base de données relationnelles existants aux spécificités du stockage d'objets, en gérant automatiquement les limites imposées sur la longueur des noms des données simples stockées dans les tables et limitant le nombre de tables annexes utilisées pour conserver de bonnes performances au système de stockage d'objets dans son ensemble.

La présente invention a donc pour objet de proposer un procédé et un système qui permettent de limiter à une table principale et une table annexe le nombre de tables nécessaire pour stocker les données simples en nombre fixe et variable d'une collection d'objets informatiques présents dans un système de stockage de données.

La présente invention propose donc un procédé de stockage d'objets informationnels ou objets dans une base de données relationnelle stockée dans un ordinateur serveur, ladite base de données relationnelle étant constituée de tables, chaque table étant constituée d'un tableau d'ensembles de données simples, lesdits ensembles ayant la même structure de données dans une même table, chaque donnée simple d'une table étant désignée par un identifiant unique dans ladite table, un objet étant constitué d'une ou plusieurs données simples pouvant être stockées dans une table de ladite base de données, et/ou d'un ou plusieurs objets emboîtés dans ledit objet, ledit emboîtement pouvant être réalisé sur un nombre quelconque de niveaux pour réaliser un objet, un objet emboîté ou une donnée simple étant dit localement en nombre fixe s'il ou elle apparaît exactement une fois dans l'objet le ou la contenant immédiatement et étant dit localement en nombre variable dans le cas contraire, une donnée simple apparaissant à un niveau quelconque dudit objet étant dite globalement en nombre fixe si elle est localement en nombre fixe et si tous les objets la contenant sont localement en nombre fixe, une donnée simple apparaissant à un niveau quelconque dudit objet étant dite globalement en nombre variable si elle est localement en nombre variable ou si l'un quelconque des objets la contenant est localement en nombre variable, lesdites données globalement en nombre fixe d'un objet à stocker étant stockées dans un ensemble de données principal stocké dans une table principale de ladite base de données, lesdites données simples globalement en nombre variable dudit objet à stocker étant stockées dans une ou plusieurs tables annexes de ladite base de données, qui a pour caractéristique le fait que, lorsqu'elles existent, les données simples globalement en nombre variable desdits objets sont stockées dans une unique table annexe de ladite base de données, le procédé créant

un ou plusieurs ensembles de données annexes pour stocker lesdites données simples globalement en nombre variable dans ladite unique table annexe.

Dans ce procédé, chacun desdits un ou plusieurs ensembles annexes éventuellement stockés dans ladite unique table annexe peut comporter en outre un ensemble d'indicateurs booléens, chaque indicateur booléen étant associé à une donnée particulière desdites données simples globalement en nombre variable stockées dans ledit ensemble annexe, ledit indicateur booléen indiquant si ladite donnée simple globalement en nombre variable associée est définie ou non dans ledit ensemble annexe. Dans ce cas, certains desdits indicateurs booléens dudit ensemble d'indicateurs booléens pourront être communs à plusieurs données simples globalement en nombre variable lorsque lesdites données simples sont localement en nombre fixe dans un même objet les contenant immédiatement.

De préférence, ledit ensemble principal associé au dit objet à stocker comportera une donnée permettant d'identifier de façon unique ledit ensemble principal dans ladite table principale. De plus, ladite donnée unique pourra être unique dans ledit ordinateur serveur, et avantageusement, ladite donnée unique stockée dans ledit ensemble principal associé au dit objet à stocker sera en outre stockée dans chacun des ensembles annexes associés au dit objet à stocker, pour permettre la mise en relation dudit ensemble principal et du ou desdits ensembles annexes éventuels associés au dit objet à stocker.

De façon avantageuse, ladite donnée unique pourra être constituée du nom logique de l'ordinateur serveur sur ledit réseau, du numéro de table de ladite table principale dans ladite base de données et du numéro d'ensemble de données dudit ensemble principal dans ladite table principale, ledit nom logique de l'ordinateur serveur étant unique sur ledit réseau, ladite base de données étant unique sur ledit ordinateur serveur, ledit numéro de table de ladite table principale étant unique dans ladite base de données et ledit numéro d'ensemble de données dudit ensemble principal étant unique dans ladite table principale.

Dans le procédé, deux objets à stocker pourront être dits de même type s'ils sont constitués de données simples et d'objets emboîtés de même type, deux objets de même type ayant en commun un même identifiant et les données simples et/ou les objets emboîtés se correspondant à un niveau quelconque desdits objets de même type ayant en commun un même identifiant, unique dans l'objet les contenant immédiatement.

Dans ce cas, le procédé pourra créer un identifiant global pour tous les objets de même type et pour toutes les données simples se correspondant à un niveau

quelconque desdits objets de même type, ledit identifiant global étant ledit identifiant commun pour lesdits objets de même type, et ledit identifiant global étant obtenu, pour chacune desdites données simples se correspondant, par la concaténation des identifiants, dans les objets les contenant immédiatement, de tous les objets contenant ladite donnée simple, et de l'identifiant de ladite donnée simple dans l'objet la contenant immédiatement.

En outre, le nombre de caractères de cet identifiant global pourra être tronqué au nombre de caractères permis par ladite base de données pour l'identifiant d'une donnée stockée dans une table de ladite base de données. Egalement, une ambiguïté pouvant apparaître du fait de ladite troncature pourra être résolue en effectuant les étapes consistant à :

- remplacer le dernier caractère alphabétique de l'identifiant global par le chiffre zéro, ainsi que les éventuels chiffres le suivant ;
- augmenter d'une unité le nombre constitué par l'ensemble des chiffres apparaissant à la fin dudit identifiant global jusqu'à ce que l'ambiguïté disparaisse ou que les chiffres à la fin dudit identifiant global soient entièrement constitués de chiffres neuf ;
- répéter les étapes précédentes depuis le début lorsque les chiffres apparaissant à la fin dudit identifiant global sont entièrement constitués de chiffres neuf à l'issue de l'étape précédente.

Dans le procédé de l'invention, les objets stockés dans la base de données pourront être reçus ou transmis par l'ordinateur serveur via un réseau informatique de type Internet.

De préférence, les données simples constituant lesdits objets (6) stockés dans ladite base de données seront de type texte, et avantageusement, lesdites données simples de type texte seront des données écrites en langage XML. De plus, lesdites données écrites en langage XML pourront être conformes à une description de type XML Schema.

En outre, un ensemble de données au format XML pourra être obtenu par une traduction automatique de ladite description, ledit ensemble de données et un ensemble de données complémentaire étant à leur tour traduits automatiquement en langage SQL pour définir et gérer la structure de la base de données, et pour contrôler les échanges de données avec ladite base de données.

Par ailleurs, dans le procédé de l'invention, une partie des données simples des objets reçus par ledit réseau de type Internet pourra être supprimée à l'aide d'un

patron, ou modèle, d'objet écrit en langage XML, pour interdire la modification desdites données simples via ledit réseau.

Egalement, une partie des données simples des objets transmis via ledit réseau pourra être supprimée à l'aide d'un patron d'objet écrit en langage XML pour limiter la transmission sur ledit réseau à certaines données simples desdits objets. De plus, la suppression d'une partie des données simples contenues dans les objets permettra en outre d'optimiser les requêtes à ladite base de données.

L'invention propose en outre un système de stockage d'objets informationnels ou objets, dans une base de données relationnelle stockée par un ordinateur serveur, qui a pour caractéristique le fait qu'il met en œuvre le procédé selon l'une quelconques des revendications précédentes.

Un mode de réalisation préférentiel de l'invention va maintenant être décrit, à titre d'exemple seulement, en se référant aux dessins annexés, dans lesquels :

- la figure 1 est le schéma fonctionnel du mode de réalisation préféré du procédé de stockage d'objets selon la présente invention ;
- la figure 2 est un schéma montant l'utilisation d'une table annexe unique pour les ensembles en nombre variable dans le mode de réalisation préféré du procédé de l'invention sur un cas d'exemple ;
- la figure 3 est un schéma montant l'utilisation d'un ensemble de tables annexes pour les ensembles en nombre variables dans la technique antérieure, sur le même cas d'exemple que celui de la figure 2.

Pour permettre la comparaison, un exemple de stockage d'objet de la technique antérieure sera également décrit. Dans les deux cas, on supposera, de façon non restrictive, que le système de gestion de base de données relationnelle, ou SGBDR, utilisé est mis en œuvre à l'aide du langage standard Structured Query Language, ou SQL.

Par ailleurs, pour faciliter la compréhension, on décrira les opérations de stockage concernant un objet 6 d'exemple, tel qu'une entrée dans un annuaire comportant les informations suivantes :

Entrée annuaire :

Nom, sur 40 caractères : OTOOBE

Téléphone, sur 15 caractères : +33(0)1 44 34 85 03

Télécopie, sur 15 caractères : +33(0)1 44 34 85 01

Adresse :

Rue, sur 40 caractères : 3bis, rue du Docteur-Foucault

Ville, sur 40 caractères : 92000 NANTERRE

Adresse :

Rue, sur 40 caractères : 34, bd Haussmann

Ville, sur 40 caractères : 75009 PARIS

Contact, sur 40 caractères : M. Laurent QUEREL

Contact, sur 40 caractères : M. Jean-François QUENTIN

Contact, sur 40 caractères : M. Gilles ANDRE

Dans cette entrée, les données "Adresse" et "Contact" sont supposées pouvoir figurer en nombre variable dans ledit objet 6 "Entrée annuaire", pour prendre en compte le fait qu'une entreprise peut comporter plusieurs établissement et plusieurs personnes à contacter. Pour la clarté de l'exposé, le nombre des données en nombre variable a été limité à deux dans cet exemple, mais ce nombre ne limite en rien la portée du procédé de la présente invention.

Les objets informationnels, ou objets, dont la structure de données n'est pas fixe, permettent de prendre en compte ce type de données en nombre variable, ce qui fait toute la puissance de l'approche objet par rapport à une approche traditionnelle où la structure des données est fixe, et dans laquelle il est donc nécessaire de prévoir à l'avance un nombre maximal d'occurrences pour chaque donnée, au risque d'un coté de bloquer de façon gênante un cas particulier où un nombre d'occurrences plus important que celui prévu serait nécessaire, et d'un autre coté, de gaspiller inutilement de la place de stockage dans le cas général pour quelques cas particuliers peu fréquents.

Malheureusement, les systèmes de gestion de base de données classiques actuels ne permettent pas directement le stockage d'objets, c'est-à-dire de structures de données permettant l'emboîtement de sous-structures de données et des occurrences multiples pour les éléments de ces structures, ce qui nécessite un traitement particulier des données de l'objet 6 "Entrée annuaire" pour pouvoir le stocker dans la base de données 2.

Dans la technique antérieure, ce traitement consiste à séparer les données en nombre fixe "nom", "telephone", "telecopie" et les données en nombre variable "rue", "ville" en stockant les données en nombre fixe dans une première table principale 3, et en stockant les données, ou ensembles de données, en nombre variable dans autant d'autres tables annexes qu'il existe de données, ou d'ensembles de données, en nombre variable dans l'objet 6.

Dans la technique antérieure, l'objet 6 de l'exemple précédent est alors stocké sous la forme indiqué à la figure 3, dans laquelle :

- la table 3 est la table principale stockant les ensembles 31 de données en nombre fixe constitués des données simples "cle" 11, "nom" 12, "téléphone" 13 et "télécopie" 14 ;
- la table 4 est une première table annexe stockant les ensembles 41 constitués de la donnée simple "cle" 11 et de la donnée en nombre variable "adresse", c'est-à-dire, des données simples "rue" 15 et "ville" 16 ;
- la table 5 est une seconde table annexe stockant les ensembles de données constitués de la donnée "cle" 11 et de la donnée en nombre variable "contact" 17.

Plus précisément, dans l'exemple ci-dessus, la donnée "nom" ayant pour valeur "OTOOBE" sera stockée en 12, la donnée "téléphone" ayant pour valeur "+33(0)1 44 34 85 03" sera stockée en 13, la donnée "télécopie" ayant pour valeur "+33(0)1 44 34 85 01" sera stockée en 14, la donnée "rue" du premier sous-objet "adresse", ayant pour valeur "3bis, rue du Docteur-Foucault", sera stockée en $15_1$, la donnée "ville" du premier sous-objet "adresse", ayant pour valeur "92000 NANTERRE", sera stockée en $16_1$, la donnée "rue" du second sous-objet "adresse", ayant pour valeur "34, bd Haussmann", sera stockée en $15_2$, la donnée "ville" du second sous-objet "adresse", ayant pour valeur "75009 PARIS", sera stockée en $16_2$, la première donnée "contact", ayant pour valeur "M. Laurent QUEREL", sera stockée en $17_1$, la seconde donnée "contact", ayant pour valeur "M. Jean-François QUENTIN", sera stockée en $17_2$ et la troisième donnée "contact", ayant pour valeur "M. Gilles ANDRE", sera stockée en $17_3$.

La donnée "cle" 11 apparaissant dans les trois tables 3, 4 et 5 est la clé primaire de ces tables, et son utilisation sera décrite plus loin.

Pour pouvoir utiliser ces données, il faut au préalable définir les tables 3, 4 et 5 dans la base de données 2, ainsi que les diverses données stockées dans ces tables 3, 4 et 5.

Dans la technique antérieure, pour créer la table 3, un opérateur devait typiquement indiquer manuellement au SGBDR une instruction SQL du type :

```
CREATE TABLE "EntreeAnnuaire_main" (
        cle INTEGER,
        nom VARCHAR(40),
        telephone VARCHAR(15),
        telecopie VARCHAR(15),
        PRIMARY KEY (cle))
```

Dans cette instruction, la rubrique "EntreeAnnuaire_main" indique le nom, librement choisi par l'opérateur sous réserve d'unicité, de la table principale 3, et une rubrique "VARCHAR(n)" indique une chaîne de caractères de longueur variable et de longueur maximale n. La donnée "cle" 11 est la clé primaire évoquée ci-dessus et qui sera décrite plus loin, et la rubrique "PRIMARY KEY(cle)" indique précisément au système que cette donnée "cle" 11 est la clé primaire.

De même, dans la technique antérieure, pour créer les tables 4 et 5, un opérateur devait typiquement indiquer manuellement au SGBDR des instructions SQL du type :

```
CREATE TABLE "EntreeAnnuaire_1" (
    cle INTEGER,
    rue VARCHAR(40),
    ville VARCHAR(15),
    PRIMARY KEY (cle))
```

et :

```
CREATE TABLE "EntreeAnnuaire_2" (
    cle INTEGER,
    contact VARCHAR(40),
    PRIMARY KEY (cle))
```

dans lesquelles les rubriques "EntreeAnnuaire_1" et "EntreeAnnuaire_2" sont les noms respectifs, librement choisis, des tables annexes 4 et 5.

En outre, l'opérateur devait définir également manuellement les données indexées, c'est-à-dire les données dont le contenu pourra permettre de retrouver une entrée de l'annuaire, au moyen d'instructions SQL du type :

```
CREATE INDEX "IndexNom" ON "EntreeAnnuaire_main" (nom)
```

qui indique au SGBDR de faire en sorte qu'une entrée d'annuaire puisse être retrouvée par le contenu de la rubrique "nom" de la table principale 3 "EntreeAnnuaire_main".

De plus, dans le cas d'objets 6 comportant de nombreux niveaux d'emboîtement et de nombreuses données en nombre variable comme c'est souvent le cas en pratique, le nombre de tables annexes était élevé, et la numérotation ou la dénomination des tables annexes était beaucoup plus complexe que celle présentée dans l'exemple précédent. A nouveau, cette numérotation ou cette dénomination nécessitait une intervention manuelle d'un opérateur, avec les risques d'erreur inhérents à toute opération manuelle.

En se référant aux figures 1 et 2, on va maintenant décrire le procédé de stockage selon le mode de réalisation préféré de l'invention, qui est mis en œuvre par divers modules de programmes globalement dénommés Collective Operating System ou COS dans ce document.

En revenant à l'exemple précédent, le procédé de l'invention gère l'ensemble 31 de données simples en nombre fixe "cle" 11, "nom" 12, "telephone" 13, et "telecopie" 14 dans la table principale 3 de la même manière que dans la technique antérieure, c'est-à-dire en utilisant directement les primitives adéquates du SGBDR utilisé.

Par contre, la gestion des données simple en nombre variable est effectuée à l'aide d'une unique table 7, au lieu des tables 4 et 5. Pour cela, le procédé de l'invention stocke la première occurrence d'une donnée simple en nombre variable dans un premier ensemble de données $71_1$ de la table 7, la seconde occurrence d'une donnée simple en nombre variable dans un second ensemble de données $71_2$ de la table 7, la troisième occurrence éventuelle d'une donnée simple en nombre variable dans un troisième ensemble de données $71_3$, etc.. On voit alors que la structure de données de la table 7, c'est-à-dire la structure des ensembles de données 71, doit nécessairement comporter au moins l'union des données simples en nombre variable dans un objet 6.

Comme dans la technique antérieure, la donnée "cle" 11 apparaissant dans les deux tables 3 et 7 est la clé primaire de ces tables, et son utilisation sera décrite plus loin.

En revenant à l'exemple précédent, le procédé stocke donc la donnée première occurrence de la donnée simple "rue" de l'objet 6 dans l'emplacement $15_1$ de l'ensemble $71_1$ de la table 7, il stocke la première occurrence de la donnée "ville" dans l'emplacement $16_1$ de l'ensemble $71_1$ de la table 7, il stocke la seconde occurrence de la donnée "rue" dans l'emplacement $15_2$ de l'ensemble $71_2$, il stocke la seconde occurrence de la donnée "ville" $16_2$ dans l'emplacement $16_2$ de l'ensemble $71_2$, il stocke la première occurrence de la donnée "contact" $17_1$ dans l'emplacement $17_1$ de l'ensemble $71_1$, il stocke la seconde occurrence de la donnée "contact" $17_2$ dans l'emplacement $17_2$ de l'ensemble $71_2$, et il stocke la troisième occurrence de la donnée "contact" $17_3$ dans l'emplacement $17_3$ de l'ensemble $71_3$.

Toutefois, on voit dans ce qui précède que les emplacements correspondants aux données simples "rue" et "ville" aux emplacements $15_3$ et $16_3$ dans l'ensemble de données simples $71_3$ ne sont pas remplis, ce qui est dû au fait qu'il n'y a que deux ensembles de données en nombre variable "rue" et "ville", et qu'il y a trois données

en nombre variable "contact". Cela pose un problème en ce sens que si les ensembles de données simples $71_1$, $71_2$, $71_3$ étaient stockés en l'état dans la base de données, l'information concernant quels emplacements sont remplis et quels emplacements sont vides serait longue et difficile à déterminer. Il serait en effet nécessaire de tester la nullité des colonnes de l'ensemble de ces données simples nombre variable pour déterminer l'absence ou la présence de ces données en nombre variable.

Pour résoudre ce problème, le procédé selon l'invention réservera en outre, dans chaque ensemble 71, des emplacements de données pour stocker cette information de présence ou d'absence pour chaque donnée en nombre variable dans cet ensemble 71.

En pratique, dans le cas de données simples en nombre fixe, telles que "rue" et "ville" associées dans un même sous-objet contenant tel que "adresse", les données simples en nombre fixe associées seront toujours présentes ou absentes simultanément, ce qui signifie que le procédé pourra optimiser le nombre d'indicateurs en ne stockant qu'un seul indicateur par ensemble de données en nombre fixe associées.

Dans ces conditions, lors de la création de la table 7, le procédé ajoute dans la structure de données de la table 7, c'est-à-dire dans chaque ensemble 71, une donnée booléenne pour chaque ensemble de données en nombre variable dans l'objet 6. Dans l'exemple décrit, l'ensemble de données simples 72 constitué des données simples booléennes 20 et 21 est ainsi ajouté à chaque ensemble 71, la donnée booléenne 20 indiquant la présence, dans l'ensemble 71 considéré, de l'ensemble constitué des données simples "rue" 15 et "ville" 16, et la donnée booléenne 21 indiquant la présence, dans l'ensemble 71 considéré de la donnée "contact" 17.

Ainsi, les informations de présence ou d'absence des données ou des ensembles de données en nombre variable seront conservées lors de l'écriture d'un objet 6, et le procédé pourra utiliser ces informations lors des relectures ultérieures, pour reconstituer correctement l'objet 6 tel qu'il était au moment de son enregistrement.

On notera qu'avec le procédé de l'invention, le nombre d'ensembles de donnée présents dans l'unique table annexe est égal au maximum des nombres des données en nombre variable dans l'objet 6, alors que ce nombre d'ensembles de données dans les tables annexes est égal à la somme des nombres de données en nombre variable de l'objet 6 dans la technique antérieure, c'est-à-dire un nombre toujours plus élevé.

Par ailleurs, dans le mode de réalisation préférentiel, le procédé est utilisé pour stocker des objets 6 écrits en langage XML qui sont conformes à une description 92 de type XML Schema, ce qui signifie en particulier que cette description est elle-même écrite dans le langage XML.

Cette description 92, dénommée COSSchema dans le mode de réalisation préférentiel, décrit la structure de données des objets 6. L'intérêt de cette description XML Schema 92 est de constituer une définition de référence unique pour la structure des objets 6, modèle qui sera utilisé chaque fois qu'il sera nécessaire de vérifier la structure ou la validité des objets 6.

Dans le mode de réalisation préférentiel, un certain nombre de définitions d'objets utilisées dans le schéma COSSchema 92 sont dérivées à partir de définitions stockées dans un autre schéma XML Schema 91, dénommé COSCoreSchema.

L'intérêt de cette approche est de permettre au COS d'uniformiser les définitions de certains objets en leur donnant une base commune, et de pouvoir éventuellement effectuer un certain nombre de travaux, en particulier de maintenance, de façon homogène sur ces objets. Par exemple, s'il s'avère nécessaire d'introduire une nouvelle donnée simple ou de modifier une donnée simple existante dans un type de base défini dans le COSCoreSchema, il ne sera pas nécessaire de reproduire la nouvelle donnée simple ou la modification dans tous les objets basés sur ce type, et la modification sera implicitement reproduite via la seule référence au type de base. Ainsi, les risques d'erreur au niveau de la structure des objets COS seront réduits d'autant, et la maintenance de ces objets s'en trouvera facilitée.

En revenant à l'exemple d'entrée d'annuaire vue précédemment, dans le mode de réalisation préféré de l'invention, celle-ci sera constituée d'un objet 6, se présentant en langage XML sous la forme suivante :

```
<entree_annuaire>
        <nom>OTOOBE</nom>
        <telephone>+33(0)1 44 34 85 03</telephone>
        <telecopie>+33(0)1 44 34 85 01</telecopie>
        <adresse>
            <rue>3bis, rue du Docteur-Foucault</rue>
            <ville>92000 NANTERRE</ville>
        </adresse>
        <adresse>
            <rue>34, bd Haussmann</rue>
            <ville>75009 PARIS</ville>
```

```
        </adresse>
        <contact>M. Laurent QUEREL</contact>
        <contact>M. Jean-François QUENTIN</contact>
        <contact>M. Gilles ANDRE</contact>
</entree_annuaire>
```

Cet objet sera typiquement conforme à un fragment du COSSchema, qui est de type XML Schema, se présentant sous la forme suivante :

```
<type name="entree_annuaire">
    <element name="nom" minOccurs="1">
        <datatype source="string">
            <length value="40"/>
        </datatype>
    </element>
    <element name="telephone" minOccurs="1">
        <datatype source="string">
            <length value="15"/>
        </datatype>
    </element>
    <element name="telecopie" minOccurs="1">
        <datatype source="string">
            <length value="15"/>
        </datatype>
    </element>
    <type name="adresse" minOccurs="1" maxOccurs="*">
        <element name="rue" minOccurs="1">
            <datatype source="string">
                <length value="40"/>
            </datatype>
        </element>
        <element name="ville" minOccurs="1">
            <datatype source="string">
                <length value="40"/>
            </datatype>
        </element>
    </type>
    <element name="contact" minOccurs="0" maxOccurs="*">
```

```
            <datatype source="string">
                <length value="40"/>
            </datatype>
        </element>
</type>
```

Dans cette définition XML Schema, le paramètre 'minOccurs' d'une donnée simple ou d'un sous-objet emboîté indique le nombre minimum de fois où la donnée concernée apparaît dans l'objet immédiatement contenant, et le paramètre 'maxOccurs' d'une donnée simple ou d'un sous-objet emboîté indique le nombre maximum de fois où la donnée concernée apparaît dans l'objet immédiatement contenant. Par défaut, c'est-à-dire lorsque 'maxOccurs' n'apparaît pas, il est supposé être égal à 1.

La signification de 'maxOccurs="*"' est que la donnée ou le sous-objet concerné peut apparaître un nombre quelconque de fois dans l'objet immédiatement contenant.

Dans le mode de réalisation préféré de l'invention, le procédé utilise cette description au format XML, en association avec le procédé de stockage d'objets dans seulement deux tables qui a été décrit plus haut, pour générer automatiquement des instructions SQL pour créer et gérer les données de la base de données 2 représentant les objets stockés.

Pour cela, le procédé de l'invention part du COSSchema et il applique un certain nombre de règles pour obtenir un format XML Schema étendu, dénommé XDB Schema, comparable à celui de l'exemple décrit ci-dessus. En effet, dans le COSSchema, la description des données est faite, de façon structurée, en se référant à d'autres types prédéfinis, tels que les types définis dans le COSCoreSchema ou à des types intermédiaires définis dans le COSSchema lui-même. Pour rendre utilisables dans le langage SQL utilisé les informations présentes dans le COS Schema, le procédé de l'invention, applique donc à ce schéma COS Schema un certain nombre de règles de traduction lui permettant de passer d'une représentation comportant des types non simples à une représentation ne comportant plus que des types simples. Le résultat de cette traduction constitue le schéma XDB Schema.

Toutefois, le schéma XDB Schema obtenu précédemment n'est pas suffisant à lui seul pour définir complètement la structure de l'objet 6 dans la base de données 2 : en effet, il manque les informations permettant de définir les données de l'objet 6 qui seront indexées. Pour résoudre ce problème, le procédé de l'invention utilise un

schéma XML 96 complémentaire, dénommé XDB Mapping, décrivant les données à indexer dans les tables stockant les données de l'objet 6.

Ce schéma XDB Mapping 96 reprend la structure du schéma COSSchema en la limitant aux données devant faire l'objet d'une indexation. Ainsi, dans l'exemple précédent où la donnée "nom" devait être indexée, le fragment correspondant dans le XDB Mapping sera du type suivant :

```
<type name="entree_annuaire">
    <element name="nom" sql:indexname="IndexNom" sql:indexType="..."/>
</type>
```

où la rubrique "sql:indexname" indique le nom de l'index à créer, ici "IndexNom", et où la rubrique "sql:indexType" sert à indiquer, sous une syntaxe adéquate, des paramètres pour l'index à créer, tels que "ordre de tri croissant", "pas de distinction minuscules/majuscules", etc.. Ainsi, avec ce schéma XDB Schema, le procédé de l'invention possède toutes les informations utiles pour créer automatiquement dans la base de données 2 toutes les données nécessaires pour l'objet 6 à créer.

Ensuite, le procédé traduit ces schémas XDB Schema et XDB Mapping en instructions SQL à destination du SGBD gérant la base de données 2.

Toutefois, avant de pouvoir mener à bien cette opération, un autre problème, lié aux possibilités offertes par les objets, doit être résolu. En effet, l'approche orientée objet permet à des données simples, ou à des sous-objets, contenus dans des sous-objets distincts d'avoir le même identifiant, pourvu que cet identifiant soit unique dans le sous-objet, ou l'objet 6, les contenant immédiatement. Cette possibilité de noms dupliqués dans des sous-objets différents interdit d'utiliser directement ces identifiants comme identifiants globaux de données dans la base de données 2, car deux données distinctes d'une même table pourraient alors recevoir le même identifiant, ce qui est interdit par tous les SGBDR existants.

Pour résoudre ce problème, le procédé crée pour chaque donnée simple un identifiant global obtenu en préfixant l'identifiant de la donnée simple, tel qu'il apparaît dans la rubrique 'name="..."' de la définition de la donnée simple dans le schéma XDB Schema, par la concaténation des identifiants, tels qu'il apparaissent dans les rubriques 'name="..."' des définitions correspondantes du schéma XDB Schema, de tous les sous-objets contenant, à une niveau ou un autre, la donnée simple considérée. Ce procédé lève l'ambiguïté qui pourrait apparaître, car du fait que les identifiants des données simples ou des sous-objets dans leurs objets immédiatement contenant sont distincts, deux identifiants globaux obtenus de cette façon sont nécessairement distincts globalement.

Ainsi, dans l'exemple ci-dessus, l'identifiant global de la donnée "nom" sera "nom" lui-même, puisque la donnée simple "nom" n'est pas contenue dans un sous-objet ; par contre, l'identifiant global de la donnée "rue" contenue dans le sous-objet "adresse" sera "adresse_rue".

Dans ce qui précède, on a indiqué que les identifiants des données créées dans les tables 3 et 7 de la base de données 2 était la concaténation des identifiants de la donnée et de tous les sous-objets la contenant dans l'objet 6. Toutefois, cela pose problème en ce que la longueur de ces identifiants est habituellement limitée par les SGBDR classiques à une valeur assez basse, de l'ordre de quelques dizaines de caractères, et que, pour des objets comportant un niveau d'emboîtement même modéré, cette limite est très vite atteinte.

Pour résoudre ce problème, le procédé de l'invention tronque les identifiants précédemment obtenus à la longueur permise par le SGBDR pour les identifiants de données d'une table. A l'issue de cette troncature, si l'identifiant global obtenu est déjà présent dans la table où il devait être créé, le procédé de l'invention réalise les étapes suivantes :

1°/ le dernier caractère alphabétique de l'identifiant global précédemment obtenu est remplacé par le chiffre "0", ainsi que tous les chiffres le suivant éventuellement ;

2°/ si l'identifiant global ainsi obtenu est encore présent dans la table, le nombre constitué par l'ensemble des chiffres apparaissant à la fin de l'identifiant global est augmenté d'une unité et l'étape 2 est répétée tant qu'un nombre entièrement constitué de chiffres "9" n'a pas été atteint ;

3°/ si un nombre entièrement constitué de chiffres "9" est atteint à l'étape 2, alors le procédé retourne à l'étape 1.

Les étapes 1°/ à 3°/ précédentes sont répétées jusqu'à ce qu'un identifiant global, nouveau dans la table considérée, soit trouvé.

Ainsi, le procédé ci-dessus décrit permet de lever les ambiguïtés, dans les identifiants globaux attribués aux données simples stockées dans la base de données 2, qui peuvent apparaître du fait de la possibilité de duplication d'un identifiant de donnée ou de sous-objet dans des sous-objets distincts.

En utilisant ce procédé de génération d'identifiants globaux pour les données simples d'un objet 6, le procédé relit alors en parallèle le schéma XDB Schema et XDB Mapping, et pour chaque nouveau type d'objet à créer 6 trouvé dans le schéma XDB Schema, il applique le procédé suivant :

- le procédé crée alors dans la base de données une table principale 3 portant le nom de l'objet 6, suivi du suffixe "_main", comportant une donnée "cle" 11 servant à identifier de façon unique dans la base de données un objet 6 ; ainsi, lorsqu'il rencontre dans le schéma XDB Schema la définition :

<type name="entree_annuaire">
...
</type>

le procédé crée la table principale 3 "EntreeAnnuaire_main" correspondante par l'instruction SQL :

CREATE TABLE "EntreeAnnuaire_main" (cle INTEGER, PRIMARY KEY (cle))

- ensuite, le procédé explore les définitions "<element ...>...</element>" du schéma XDB Schema, XDB Schema et pour chaque donnée simple rencontrée, telles que "nom" ou "contact", il détermine si cette donnée est en nombre fixe ou variable ; une donnée simple sera dite en nombre fixe si sa rubrique "maxOccurs" est absente de la définition et si cette donnée simple n'est pas contenue à un niveau quelconque dans un sous-objet en nombre variable, c'est-à-dire un sous-objet dont la rubrique "maxOccurs" serait présente et différente de la rubrique "minOccurs" dudit sous-objet ; une donnée sera dite en nombre variable dans le cas contraire, c'est-à-dire si elle n'est pas en nombre simple au regard de la définition précédente ;
- lorsque le procédé rencontre une donnée en nombre fixe telle que la donnée "nom" dans l'exemple ci-dessus, il ajoute cette donnée en nombre fixe dans la table principale 3 à l'aide d'une instruction SQL telle que :

ALTER TABLE "EntreeAnnuaire_main" ADD nom VARCHAR(40)

dans laquelle l'identifiant global "nom" est obtenu par le procédé décrit plus haut, et dans laquelle la longueur "40" apparaissant dans la rubrique "VARCHAR" est reprise de la '<length value="...">' de la définition correspondante ; en outre, si le procédé trouve dans le schéma XDB Mapping une définition d'index portant sur la donnée qu'il vient de créer dans la table principale, le procédé crée un index pour cette donnée à l'aide d'une instruction SQL adéquate telle que :

CREATE INDEX "IndexNom" ON "EntreeAnnuaire_main" (nom)

dans laquelle le nom de l'index "IndexNom" est celui retrouvé dans la rubrique "sql:indexname" de la définition correspondante du schéma XDB Mapping ;

- lorsque le procédé rencontre une donnée en nombre variable telle que la donnée "contact" dans l'exemple ci-dessus, il effectue les opérations suivantes :
  - s'il n'existe pas déjà une table annexe 7 associée à cette table principale 3, il crée cette table annexe à l'aide d'une instruction SQL telle que :

    CREATE TABLE "EntreeAnnuaire_annex" (cle INTEGER, PRIMARY KEY (cle))
  - dans laquelle la donnée "cle" est la donnée servant à identifier de façon unique un objet 6 dans la base de données 2 ;
  - ensuite, le procédé crée la donnée correspondante dans la seconde table, en lui associant, de la façon décrite précédemment, une donnée booléenne "presence", permettant de savoir si la donnée correspondante est présente ou non dans l'ensemble de données 71 considéré ; par exemple, pour la donnée en nombre variable "contact" 17, cette création sera réalisée à l'aide d'une instruction SQL telle que :

    ALTER TABLE "EntreeAnnuaire_annex" ADD contact VARCHAR(40), ADD presence_contact BOOLEAN
  - dans laquelle l'identifiant global est obtenu par le procédé décrit plus haut, et dans laquelle longueur "40" apparaissant dans la rubrique "VARCHAR" est reprise de la rubrique '<length value="...">' de la définition correspondante dans le schéma XDB Schema ;
  - dans le cas d'une donnée incluse dans un sous-objet comme la donnée "rue", l'identifiant global sera obtenu comme indiqué précédemment, et l'instruction SQL correspondante sera, par exemple :

    ALTER TABLE "EntreeAnnuaire_annex" ADD adresse_rue VARCHAR(40), ADD presence_adresse_rue BOOLEAN.

Le procédé répète les opérations précédentes jusqu'à ce que toutes les définitions présentes dans le schéma XDB Schema 61 aient été traitées. A l'issue de

ce traitement, le procédé aura donc créé automatiquement la structure d'objets 6 dans la base de données 2, telle qu'elle était décrite dans le schéma COS Schema.

Dans le mode de réalisation de la présente invention, les aspects ci-dessus du procédé de l'invention sont implémentés dans un programme 90 dénommé XBD Engine, et, contrairement à la technique antérieure, le procédé de l'invention permet de réaliser la création de nouveau type d'objet 6 dans la base de données 2 sans intervention manuelle d'un opérateur, ce qui se traduit par un gain de temps très appréciable et une diminution considérable du risque d'erreur, tant pour la création que pour la modification, dans la base de données, de la structure d'un objet 6.

Dans ce qui précède, on a décrit le procédé de l'invention comme transformant le schéma COSSchema, comportant des types prédéfinis, en un schéma XML, le schéma XDB Schema, ne comportant plus que des types de données simples tels qu'utilisés par le SGBDR. Cela a été fait dans l'optique de l'utilisation d'un langage SQL de base, qui ne connaît pas de types autres que ces types simples.

Toutefois, dans le cas de l'utilisation d'un langage SQL évolué, tel que le langage SQL 3, le procédé de l'invention tirera naturellement parti des possibilités de typage offertes par ce langage en ne traduisant en types plus simples que les types ne pouvant pas être "compris" directement par le langage SQL utilisé. Les types utilisés par le COS Schema et pouvant être traduits directement en types SQL, le seront à l'aide d'une instruction SQL adéquate, telle que :

CREATE TYPE ...

Dans ce qui précède, on a décrit sur un exemple les étapes de création d'un nouveau type d'objet 6 à stocker dans la base de données 2, tant dans la technique antérieure que dans le procédé de l'invention. Les informations contenues dans COSSchema et XDBSchema seront bien évidemment réutilisées pour modifier la structure d'un objet 6 déjà existant dans la base de données 2, après que les modifications voulues y aient été apportées. Cette modification de la structure d'un type d'objet 6 déjà existant est en tout point similaire à la création précédemment décrite. Ainsi, par exemple, pour modifier une table 3 existante et y ajouter un nouveau champ, l'instruction SQL utilisée, tant dans la technique antérieure que dans le procédé de l'invention :

CREATE TABLE ...

sera remplacée, par exemple, par l'instruction SQL :

ALTER TABLE "EntreeAnnuaire_main" ADD telex VARCHAR(15),
DROP telecopie

qui indique qu'une donnée "telex" sur quinze caractères doit être ajoutée et que la donnée "telecopie" doit être supprimée.

De même, la suppression de l'index sur la donnée "nom" dans la table 3 se ferait, tant dans la technique antérieure que dans le procédé de l'invention, par l'instruction SQL :

DROP INDEX EntreeAnnuaire.IndexNom

Dans le cas du procédé de l'invention, à nouveau ces modifications sont effectuées de façon automatique par le programme XDB Engine.

Compte-tenu de ces similitudes, la modification de la structure d'un objet 6 stocké dans la base de données 2 par le procédé de l'invention ne sera pas décrite plus en détail.

Dans ce qui suit, on va maintenant décrire le fonctionnement du procédé selon la présente invention en comparaison avec la technique antérieure pour le stockage et la gestion des données en nombre variable d'un objet 6.

En se référant aux figures 2 et 3, un système de stockage des données simples d'un objet 6, dans la technique antérieure comme dans le procédé de l'invention, est constitué d'un ordinateur 1 hébergeant une base de données relationnelle 2. Les ensembles de données simples d'une table principale telle que la table principale 3, ou ceux d'une table annexe telles que les tables 4, 5 ou 7, sont gérés à l'aide des fonctions offertes par le système de gestion de base de données relationnelle ou SGBDR, utilisé pour la gestion de la base de données 2.

Ces fonctions comprennent au minimum une fonction d'écriture ou de création d'un ensemble de données simples dans une table, une fonction de lecture d'un ensemble de données simples depuis une table, une fonction de suppression d'un ensemble d'une table, et une fonction de recherche d'un ensemble par son contenu, la fonction de modification d'un ensemble pouvant être assimilée, pour la simplicité de l'exposé, à une lecture de l'ensemble suivie d'une écriture de l'ensemble modifié. Par conséquent, cette fonction ne sera pas décrite davantage.

Dans ce qui suit, on va comparer les performances, en termes de nombres d'ensembles de données échangés avec la base de données 2, respectivement pour la technique antérieure et le procédé de l'invention.

En se référant à la figure 3, dans la technique antérieure, la base de données 2 comporte une table principale 3 stockant les ensembles 31 de données simples en nombre fixe "nom" 12, "telephone" 13 et "telecopie" 14, et cette table principale est en relation avec deux tables annexes 4 et 5 contenant les ensembles 41 et 51 de données simples en nombre variable, respectivement les ensembles 41 constitués des

données simples "rue" 15 et "ville" 16 pour la table annexe 4, et les ensembles 51 constitués de la donnée "contact" 17 pour la table annexe 5.

De façon classique, une donnée d'identification unique "cle" 11, présente dans les ensembles 31, 41 et 51 respectivement stockés dans les tables 3, 4 et 5, est utilisée pour mettre en relation, c'est-à-dire faire se référencer, la table principale 3 et les tables annexes 4 et 5.

1°/ Ecriture d'un objet 6 dans la base de données 2

L'écriture, ou création, d'un objet identifié par une valeur unique telle que 1234 et comportant les données simples en nombre fixe "nom" 12, "telephone" 13, "telecopie" 14, associées aux données simples "rue" 15, "ville" 16 et "contact" 17 en nombre variable, comportera classiquement les étapes suivantes :

- écriture de l'ensemble principal 31 constitué des données simples "cle" 11, "nom" 12, "telephone" 13 et "telecopie" 14 dans la table 3 à l'aide de la fonction correspondante du SGBDR, en initialisant la donnée "cle" 11 à la valeur 1234 ; cela pourra être réalisé, par exemple, à l'aide d'une instruction SQL telle que :

  INSERT INTO "EntreeAnnuaire_main" (cle, nom, telephone, telecopie) VALUES (1234, "OTOOBE", "+33(0)1 44 34 85 03", "+33(0)1 44 34 85 01")

- écriture de deux ensembles 41 constitués des données simples en nombre variable 15 et 16 à l'aide de la fonction correspondante du SGBDR, en initialisant dans chacun la donnée "cle" 11 à la valeur 1234, c'est-à-dire, écriture des deux ensembles $41_1$ et $41_2$ constitués respectivement des données simples "cle" 11, "rue" $15_1$, "ville" $16_1$ d'une part, et des données simples "cle" 11, "rue" $15_2$, "ville" $16_2$ d'autre part ; cela pourra être réalisé, par exemple, par les instructions SQL :

  INSERT INTO "EntreeAnnuaire_1" (cle, rue, ville) VALUES (1234, "3bis, rue du Docteur-Foucault", "92000 NANTERRE")

  et :

  INSERT INTO "EntreeAnnuaire_1" (cle, rue, ville) VALUES (1234, "34, bd Haussmann", "75009 PARIS")

- écriture de trois ensembles 51 à l'aide de la fonction correspondante du SGBDR, en initialisant la donnée "cle" 11 à la valeur 1234, c'est-à-dire, écriture des trois ensembles $51_1$, $51_2$ et $51_3$ constitués respectivement des données simples "cle" 11 et "contact" $17_1$ pour le premier, des données simples "cle" 11 et "contact" $17_2$ pour le second, et "cle" 11 et "contact"

$17_1$ pour le troisième ; cela pourra être réalisé, par exemple, par les instructions SQL :

INSERT INTO "EntreeAnnuaire_2" (cle, contact) VALUES (1234, "M. Laurent QUEREL")

INSERT INTO "EntreeAnnuaire_2" (cle, contact) VALUES (1234, "M. Jean-François QUENTIN")

et :

INSERT INTO "EntreeAnnuaire_2" (cle, contact) VALUES (1234, "M. Gilles ANDRE")

On constate que, dans la technique antérieure, on effectue donc une opération d'écriture pour stocker l'ensemble 31, deux opérations d'écriture pour stocker les ensembles 41, et trois écritures pour stocker les ensembles 51, soit un total de six écritures pour stocker l'ensemble des données correspondant à l'objet à stocker 6.

2°/ Lecture d'un objet 6 dans la base de données 2

La lecture d'un objet 6, ayant pour donnée unique "cle" 11 une valeur donnée telle que 1234, et comportant les données simples en nombre fixe "nom" 12, "telephone" 13, "telecopie" 14, associées aux données simples "rue" 15, "ville" 16 et "contact" 17 en nombre variable, se fera classiquement par les étapes suivantes :

- lecture, à l'aide de la fonction correspondante du SGBDR, de l'ensemble principal 31, constitué des données simples "cle" 11, "nom" 12, "telephone" 13 et "telecopie" 14, dont la donnée "cle" 11 a la valeur donnée 1234 ; cela pourra être réalisé, par exemple, à l'aide de l'instruction SQL :

  SELECT * FROM "EntreeAnnuaire_main" WHERE cle=1234

- lecture, l'aide de la fonction correspondante du SGBDR, de tous les ensembles 41 de la table 4 comportant la valeur 1234 dans leur donnée "cle" 11, c'est-à-dire, lecture des deux ensembles $41_1$ et $41_2$ constitués respectivement des données simples "cle" 11, "rue" $15_1$, "ville" $16_1$ d'une part, et des données simples "cle" 11, "rue" $15_2$, "ville" $16_2$ d'autre part ; cela pourra être obtenu, par exemple, à l'aide de l'instruction SQL :

  SELECT * FROM "EntreeAnnuaire_1" WHERE cle=1234

  cette instruction retrouvera alors les deux ensembles de données $41_1$ et $41_2$ ;

- lecture des ensembles 51 à l'aide de la fonction correspondante du SGBDR, en relisant tous les ensembles 51 de la table 5 comportant la

valeur donnée 1234 dans leur donnée "cle" 11, c'est-à-dire, lecture des trois ensembles $51_1$, $51_2$ et $51_3$ constitués respectivement des données simples "cle" 11 et "contact" $17_1$ pour le premier, des données simples "cle" 11 et "contact" $17_2$ pour le second, et "cle" 11 et "contact" $17_1$ pour le troisième ; cela pourra être obtenu, par exemple, à l'aide de l'instruction SQL :

SELECT * FROM "EntreeAnnuaire_2" WHERE cle=1234

cette instruction retrouvera alors les trois ensembles de données $51_1$, $51_2$ et $51_3$.

De même que pour l'opération d'écriture, on constate que, dans la technique antérieure, on effectue donc au total de six opérations de lecture d'ensemble de données dans les tables 3, 4 et 5 pour relire l'ensemble des données correspondant à l'objet 6 à relire.

3°/ Suppression d'un objet 6 dans la base de données 2

La suppression d'un objet, constitué des données simples "cle" 11, "nom" 12, "telephone" 13 et "telecopie" 14, ayant pour donnée "cle" 11 une valeur donnée, par exemple 1234, et comportant les données simples en nombre fixe "nom" 12, "telephone" 13, "telecopie" 14, associées aux données simples en nombre variable "rue" 15, "ville" 16 et "contact" 17, se composera classiquement des étapes suivantes :

- suppression de l'ensemble principal 31 ayant pour donnée "cle" 11 la valeur donnée 1234 à l'aide de la fonction correspondante du SGBDR ; cela pourra être réalisé, par exemple, à l'aide de l'instruction SQL :
  DELETE FROM "EntreeAnnuaire_main" WHERE cle=1234
- suppression des ensembles 41 ayant comme donnée "cle" 11 la valeur donnée 1234 à l'aide de la fonction correspondante du SGBDR, c'est-à-dire, suppression des deux ensembles $41_1$ et $41_2$ constitués respectivement des données simples "cle" 11, "rue" $15_1$, "ville" $16_1$ d'une part, et des données simples "cle" 11, "rue" $15_2$, "ville" $16_2$ d'autre part ; cela pourra être réalisé, par exemple, à l'aide de l'instruction SQL :
  DELETE FROM "EntreeAnnuaire_1" WHERE cle=1234
- suppression des ensembles 51 ayant comme donnée "cle" 11 la valeur donnée 1234 à l'aide de la fonction correspondante du SGBDR, c'est-à-dire, suppression des trois ensembles $51_1$, $51_2$ et $51_3$ constitués respectivement des données simples "cle" 11 et "contact" $17_1$ pour le premier, des données simples "cle" 11 et "contact" $17_2$ pour le second, et

"cle" 11 et "contact" $17_3$ pour le troisième ; cela pourra être réalisé, par exemple, à l'aide de l'instruction SQL :

DELETE FROM "EntreeAnnuaire_2" WHERE cle=1234

De même que pour les opérations d'écriture ou de lecture, on constate que, dans la technique antérieure, on effectue au total de six opérations de suppression d'ensemble de données dans les tables 3, 4, et 5 pour supprimer l'ensemble des données correspondant à l'objet 6 à supprimer.

4°/ Recherche d'un objet 6 dans la base de données 2

Toujours dans la technique antérieure, la recherche d'un objet ayant, par exemple, pour donnée 12 "nom" la valeur "OTOOBE", et comportant un ensemble 31 de données simples en nombre fixe "nom" 12, "telephone" 13, "telecopie" 14 associées aux données simples "rue" 15, "ville" 16 et "contact" 17 présentes dans les ensembles 41 et 51, sera constituée classiquement des étapes suivantes :

- recherche, à l'aide de la fonction correspondante du SGBDR, de l'ensemble principal 31, constitué des données simples "cle" 11, "nom" 12, "telephone" 13 et "telecopie" 14, dont la donnée 12 "nom" possède la valeur "OTOOBE" donnée ; cela pourra être réalisé, par exemple, à l'aide de l'instruction SQL :

  SELECT * FROM "EntreeAnnuaire_main" WHERE nom="OTOOBE"

- lecture, à l'aide de la fonction correspondante du SGBDR, de tous les ensembles 41 de la table 4 ayant, dans leur donnée "cle" 11, la valeur contenue dans la donnée "cle" 11 de l'ensemble 31 venant d'être retrouvé dans la table 3, c'est-à-dire, lecture des deux ensembles $41_1$ et $41_2$ constitués respectivement des données simples "cle" 11, "rue" $15_1$, "ville" $16_1$ d'une part, et des données simples "cle" 11, "rue" $15_2$, "ville" $16_2$ d'autre part ; en supposant que la valeur contenue dans la donnée "cle" 11 de l'ensemble 31 retrouvé précédemment soit égale à 1234, cela pourra être obtenu, par exemple, à l'aide de l'instruction SQL :

  SELECT * FROM "EntreeAnnuaire_1" WHERE cle=1234

- lecture, à l'aide de la fonction correspondante du SGBDR, de tous les ensembles 51 de la table 5 ayant dans leur donnée "cle" 11, la valeur contenue dans la donnée "cle" 11 de l'ensemble 31 venant d'être retrouvé dans la table 3, c'est-à-dire, lecture des trois ensembles $51_1$, $51_2$ et $51_3$ constitués respectivement des données simples "cle" 11 et "contact" $17_1$ pour le premier, des données simples "cle" 11 et "contact" $17_2$ pour le second, et "cle" 11 et "contact" $17_1$ pour le troisième ; en supposant

toujours que la valeur contenue dans la donnée 1 "cle" de l'ensemble 31 retrouvé soit égale à 1234, cela pourra être obtenu, par exemple, à l'aide de l'instruction SQL :

SELECT * FROM "EntreeAnnuaire_2" WHERE cle=1234

De même que pour les opérations d'écriture, de lecture ou de suppression, on constate que, dans la technique antérieure, on relit au total de six ensembles de données depuis les tables 3, 4 et 5 pour retrouver l'ensemble des données correspondant à l'objet 6 recherché.

En se référant maintenant à la figure 2, dans le procédé de l'invention, la base de données 2 comporte une table principale 3 stockant les ensembles 31 de données simples en nombre fixe "cle" 11, "nom" 12, "telephone" 13 et "telecopie" 14, et cette table principale est en relation avec une unique table annexe 7 contenant les ensembles 71 de données simples en nombre variable, constitués des données simples "contact" 17. De façon classique, une donnée d'identification unique "cle" 11, présente dans les ensembles 31 et 71 respectivement stockés dans les tables 3 et 7, est utilisée pour mettre en relation, c'est-à-dire faire se référencer, la table principale 3 et la table annexe 7, et pour identifier de façon unique un objet 6 dans la base de données 2.

1°/ Ecriture d'un objet 6 dans la base de données 2

L'écriture, ou création, d'un objet identifié par une valeur unique telle que 1234 et comportant les données simples en nombre fixe "nom" 12, "telephone" 13, "telecopie" 14, associées aux données simples "rue" 15, "ville" 16 et "contact" 17 en nombre variable, comportera, dans le procédé de l'invention, les étapes suivantes :

- écriture de l'ensemble d'informations principal 31, constitué des données simples "cle" 11, "nom" 12, "telephone" 13 et "telecopie" 14, à l'aide de la primitive adéquate du SGBDR, ce qui conduit à une opération d'écriture pour cet ensemble 31 ; en supposant toujours que la valeur pour la donnée "cle" 11 soit 1234, cela pourra être réalisé à l'aide d'une instruction SQL telle que :

  INSERT INTO "EntreeAnnuaire_main" (cle, nom, telephone, telecopie) VALUES (1234, "OTOOBE", "+33(0)1 44 34 85 03", "+33(0)1 44 34 85 01")

- stockage de la valeur 1234 dans les données "cle" 11 des trois ensembles de données simples $71_1$, $71_2$, $71_3$, stockage de la première donnée "rue" dans la donnée $15_1$ de l'ensemble $71_1$ de la table 7, stockage de la première donnée "ville" dans la donnée $16_1$ de l'ensemble $71_1$, stockage

de seconde donnée "rue" dans la donnée $15_2$ de l'ensemble $71_2$, stockage de la seconde donnée "ville" dans la donnée $16_2$ de l'ensemble $71_2$, stockage de la première donnée "contact" dans la donnée $17_1$ de l'ensemble $71_1$, stockage de la seconde donnée "contact" dans la donnée $17_2$ de l'ensemble $71_2$, et stockage de la troisième donnée "contact" dans la donnée $17_3$ de l'ensemble $71_3$ ;

- initialisation de la donnée $20_1$ à la valeur booléenne VRAI puisque les données simples "rue" $15_1$ et "ville" $16_1$ correspondantes sont présentes dans l'ensemble $71_1$, initialisation de la donnée $21_1$ à la valeur VRAI puisque la donnée "contact" $17_1$ correspondante est présente dans l'ensemble $71_1$, initialisation de la donnée $20_2$ à la valeur VRAI puisque les données simples "rue" $15_2$ et "ville" $16_2$ correspondantes sont présentes dans l'ensemble $71_2$, et initialisation de la donnée $21_2$ à la valeur VRAI puisque la donnée "contact" $17_2$ correspondante est présente dans l'ensemble $71_2$; par contre, initialisation de la donnée $20_3$ à la valeur FAUX puisque les données simples "rue" $15_3$ et "ville" $16_3$ correspondantes sont absentes de l'ensemble $71_3$ ; enfin, initialisation de la donnée $21_3$ à la valeur VRAI puisque la donnée "contact" $17_3$ correspondante est bien présente dans l'ensemble $71_3$ ;
- écriture les trois ensembles $71_1$, $71_2$, $71_3$ dans la base de données 2 à l'aide d'instructions SQL telles que :

```
INSERT INTO "EntreeAnnuaire_annex" (cle, presence_adresse,
    presence_contact, rue, ville, contact) VALUES (1234, TRUE,
    TRUE, "3bis, rue du Docteur-Foucault", "92000 NANTERRE",
    "M. Laurent QUEREL")
INSERT INTO "EntreeAnnuaire_annex" (cle, presence_adresse,
    presence_contact, rue, ville, contact) VALUES (1234, TRUE,
    TRUE, "34, bd Haussmann", "75009 PARIS", "M. Jean-François
    QUENTIN")
```

et :

```
INSERT INTO "EntreeAnnuaire_annex" (cle, presence_adresse,
    presence_contact, rue, ville, contact) VALUES (1234, FALSE,
    TRUE, "", "M. Gilles ANDRE")
```

Ainsi, dans le procédé selon l'invention, le nombre d'écritures dans la base de données 2 est égal à quatre, au lieu de six dans la technique antérieure.

2°/ Relecture d'un objet 6 dans la base de données 2

La lecture d'un objet 6, ayant pour donnée unique "cle" 11 une valeur donnée telle que 1234, et comportant les données simples en nombre fixe "nom" 12, "telephone" 13, "telecopie" 14, associées aux données simples "rue" 15, "ville" 16 et "contact" 17 en nombre variable, se fera, dans le procédé de l'invention, par les étapes suivantes :

- lecture, à l'aide de la fonction correspondante du SGBDR, de l'ensemble principal 31, constitué des données simples "cle" 11, "nom" 12, "telephone" 13 et "telecopie" 14, dont la donnée "cle" 11 a la valeur donnée 1234, ce qui conduit à une opération de lecture pour cet ensemble ; cette opération de lecture pourra être réalisée, par exemple, à l'aide de l'instruction SQL :

  SELECT * FROM "EntreeAnnuaire_main" WHERE cle=1234

  à l'aide des données présentes dans l'ensemble 31, le procédé de l'invention restitue les données fixes de l'objet 6 ; plus précisément, le procédé initialise la donnée en nombre fixe "nom" de l'objet 6 à l'aide de la donnée 12 présente dans l'ensemble 31 relu, il initialise la donnée en nombre fixe "telephone" de l'objet 6 à l'aide de la donnée 13 de ce même ensemble, et il initialise la donnée en nombre fixe "telecopie" de l'objet 6 à l'aide de la donnée 14 ;
- lecture, dans la table 7, des ensembles 71 de cette table comportant la valeur 1234 dans leur donnée "cle" 11 ; cela pourra être réalisé à l'aide d'un instruction SQL telle que :

  SELECT * FROM "EntreeAnnuaire_annex" WHERE cle=1234

  les trois ensembles de données $71_1$, $71_2$, $71_3$ sont ainsi relus, moyennant trois opérations de lecture d'ensemble de données dans la base de données 2 ;

  ainsi que cela a été décrit précédemment, ces trois ensembles $71_1$, $71_2$, $71_3$ de données sont chacun constitués de la donnée "cle" 11, des données simples en nombre variable "rue" 15, "ville" 16 et "contact" 17, et des données 20 et 21 indiquant respectivement la présence ou non de l'ensemble en nombre variable "rue", "ville" et de la donnée en nombre variable "contact" dans l'ensemble 71 concerné ;
- reconstitution des données variables de l'objet 6 à l'aide des données présentes dans les ensembles de données $71_1$, $71_2$, $71_3$ précédemment relus ; pour cela, le procédé examine les données 20 et 21 dans chaque ensemble $71_1$, $71_2$, $71_3$ relu ;

- plus précisément, si le contenu de la donnée booléenne $20_1$ contenue dans l'ensemble $71_1$ est VRAI, cela signifie qu'il existait à l'origine un premier ensemble "rue", "ville" dans l'objet 6 stocké ; dans ce cas, le procédé selon l'invention créera alors un premier sous-objet en nombre variable "adresse" dans l'objet 6 ; il remplira les données en nombre variable "rue" et "ville" de ce premier sous-objet avec respectivement les données $15_1$ et $16_1$ contenues dans l'ensemble $71_1$ ; de même, si le contenu de la donnée booléenne $21_1$ est vrai, cela signifie qu'il existait à l'origine une première donnée en nombre variable "contact" dans l'objet 6 ; dans ce cas, le procédé de l'invention créera une première donnée en nombre variable "contact" dans l'objet 6 ;
- en utilisant la même démarche que ci-dessus, le procédé de l'invention créera un second et un troisième sous-objet "adresse" en nombre variable "rue", "ville" si les contenus des données booléennes $20_2$ et $20_3$ contenues respectivement dans les ensembles $71_2$ et $71_3$ sont VRAI, et il initialisera les données "rue" et "ville" de ces second et troisième sous-objets en nombre variable "adresse" avec les données respectivement $15_2$ et $16_2$ d'une part, et $15_3$ et $16_3$ d'autre part respectivement contenues dans l'ensemble $71_2$ et l'ensemble $71_3$ ;
- de même, le procédé de l'invention créera une seconde et une troisième donnée en nombre variable "contact" si les contenus des données booléennes $21_2$ et $21_3$ contenues respectivement dans les ensembles $71_2$ et $71_3$ sont VRAI, et il initialisera ces seconde et troisièmes données en nombre variable "contact" avec les données $17_2$ et $17_3$ respectivement contenues dans l'ensemble $71_2$ et l'ensemble $71_3$ ;
- en supposant que les données relues proviennent de l'exemple d'écriture d'objet 6 précédemment décrit, les données simples $20_1$, $20_2$, $21_1$, $21_2$ et $21_3$ seront à la valeur VRAI, tandis que la donnée $20_3$ sera à la valeur FAUX ; dans ces conditions, le procédé de l'invention reconstituera exactement deux ensembles de données "adresse" en nombre variable dans l'objet 6 relu, ainsi que trois données en nombre variable "contact" ; l'objet 6 d'origine sera donc parfaitement reconstitué.

Pour relire l'objet 6, le procédé effectue donc au total trois opérations de lecture d'ensembles de données dans la base de données 2, contre six pour la technique antérieure.

3°/ Suppression d'objet 6 dans la base de données 2

La suppression d'un objet ayant pour donnée "cle" 11 une valeur donnée, par exemple 1234, et comportant les données simples en nombre fixe "nom" 12, "telephone" 13, "telecopie" 14, associées aux données simples en nombre variable "rue" 15, "ville" 16 et "contact" 17, se composera, dans le procédé de l'invention, des étapes suivantes :

- suppression de l'ensemble principal 31, constitué des données simples "cle" 11, "nom" 12, "telephone" 13 et "telecopie" 14, ayant pour donnée "cle" 11 la valeur donnée 1234, à l'aide de la fonction correspondante du SGBDR ; cela pourra être réalisé, par exemple, à l'aide de l'instruction SQL :
  DELETE FROM "EntreeAnnuaire_main" WHERE cle=1234
- suppression de l'ensemble annexe 71, ayant pour donnée "cle" 11 la valeur donnée 1234, à l'aide de la fonction correspondante du SGBDR ; cela pourra être réalisé, par exemple, à l'aide de l'instruction SQL :
  DELETE FROM "EntreeAnnuaire_annex" WHERE cle=1234

A nouveau, le nombre d'opérations de suppression d'ensembles de données par le SGBDR s'élève à un total de quatre dans le procédé de l'invention, contre six dans la technique antérieure.

4°/ Recherche d'un objet 6 dans la base de données 2

- recherche, à l'aide de la fonction correspondante du SGBDR, de l'ensemble principal 31, constitué des données simples "cle" 11, "nom" 12, "telephone" 13 et "telecopie" 14, dont la donnée 12 "nom" possède la valeur "OTOOBE" donnée ; cela pourra être réalisé, par exemple, à l'aide de l'instruction SQL :
  SELECT * FROM "EntreeAnnuaire_main" WHERE nom="OTOOBE"
- lecture, à l'aide de la fonction correspondante du SGBDR, de tous les ensembles 71 de la table 7 ayant, dans leur donnée "cle" 11, la valeur contenue dans la donnée "cle" 11 de l'ensemble 31 venant d'être retrouvé dans la table 3, c'est-à-dire, lecture des trois ensembles $71_1$, $71_2$ et $71_3$ ayant la structure de données précédemment décrite ; en supposant que la valeur contenue dans la donnée "cle" 11 de l'ensemble 31 retrouvé

précédemment soit égale à 1234, cela pourra être obtenu, par exemple, à l'aide de l'instruction SQL :

SELECT * FROM "EntreeAnnuaire_annex" WHERE cle=1234

- reconstitution des données ou des sous-objets en nombre variable de l'objet 6 à relire à l'aide des données contenues dans les trois sous-ensembles relecture ; cette reconstitution est identique à celle décrite au paragraphe 2°/ ci-dessus ; en conséquence, elle ne sera pas décrite plus en détail.

A nouveau, le nombre d'opérations de recherche d'ensembles de données par le SGBDR s'élève à un total de quatre dans le procédé de l'invention, contre six dans la technique antérieure.

Ainsi, dans les quatre opérations de base effectuées par un SGBDR, l'utilisation du procédé de l'invention permet une réduction d'un tiers du nombre d'opérations d'ensembles de données dans tous les cas. L'exemple présenté ci-dessus, volontairement simple pour des raisons de clarté de l'exposé, ne rend pas nécessairement bien compte des gains très importants que peut apporter le procédé dans un cas pratique comportant un nombre tables en relation beaucoup plus important.

Dans un cadre plus général, on désigne, dans ce qui suit, par $N_a$ le nombre de tables annexes $T_i$, par $N_p$ le nombre total d'ensembles principaux 31 dans la table principale 3, et par $K_i$, pour i variant de 1 à $N_a$, le nombre total d'ensembles de données présents dans une table annexe $T_i$.

Avec ces conventions, dans la technique antérieure, le nombre moyen d'opérations pour effectuer une entrée-sortie concernant un objet 6 dans une table $T_i$ donnée sera, en moyenne, égal à $K_i/N_a$, et le total général pour une entrée-sortie complète comprenant l'ensemble principal 31 et les $N_a$ ensembles de données présents dans les $N_a$ tables annexes $T_i$ sera donc égal à :

$$N_{ant} = 1 + \sum_{i=1}^{Na} K_i/N_p$$

Dans le cas du procédé de la présente invention, le nombre d'entrées-sorties nécessaire pour obtenir les ensembles de données contenus dans l'unique table annexe sera donc égal en moyenne à $\max_{i=1} (K_i)/N_p$, du fait que l'unique table annexe comporte un nombre d'ensembles égal au maximum des nombres d'ensembles présents dans les ensembles d'ensembles en relation ; par conséquent, le total général

pour une entrée-sortie complète comprenant l'ensemble principal 31 et les ensembles de données contenus dans l'unique table annexe sera donc égal à :

$$N_{proc} = 1 + \max_{i=1}^{Na} (K_i)/N_p$$

On voit que sauf circonstance très exceptionnelle, le nombre moyen d'entrées-sorties $N_{proc}$ avec le procédé de la présente invention sera toujours très inférieur au nombre moyen d'entrées-sorties $N_{ant}$ sans ce procédé. Dans le cas favorable, mais relativement fréquent, où le nombre d'ensembles de données est sensiblement le même dans les diverses tables annexes, le procédé selon l'invention permet un gain en performances atteignant un facteur égal au rapport $N_{proc}/N_{ant}$, c'est-à-dire :

$$\frac{1 + \max_{i=1}^{Na} (K_i)/N_p}{1 + \sum_{i=1}^{Na} K_i/N_p}$$

soit en utilisant l'hypothèse que les divers $K_i$ sont sensiblement égaux entre eux, et donc au maximum d'entre eux :

$$\frac{1 + \max_{i=1}^{Na} (K_i)/N_p}{1 + N_a \times \max_{i=1}^{Na} (K_i)/N_p}$$

soit en négligeant 1 devant $\max_{i=1}^{Na} (K_i)/N_p$, un rapport de valeur $1/N_a$, ce constitue une diminution très importante du nombre d'entrées-sorties dès que le nombre de tables annexes augmente.

Ce procédé est donc particulièrement souhaitable dans toutes les applications de bases de données relationnelles dans lesquelles les performances constituent un facteur critique, telles que les applications temps réel ou celles utilisant des serveurs de bases de données très sollicités, comme ceux qui se rencontrent dans les réservations centralisées par un réseau, ou dans la consultation de bases de données via le réseau mondial Internet.

En outre, l'utilisation du langage XML pour représenter les objets stockés dans la base de données 2 permet l'utilisation de patrons d'objets XML pour filtrer certaines données des objets 6 entrants et sortants de la base de données 2. Ainsi, dans le mode de réalisation préféré de la présente invention, des patrons d'objets

XML 61, collectivement dénommés I/O Format et conformes à une description XML Schema, sont utilisés pour réaliser un tel filtrage.

Plus précisément, ces patrons d'objets XML I/O Format 61 supprimeront certaines données sensibles des objets entrants, comme des données concernant la validité des données contenues dans un objet 6, de façon à ce que ces données sensibles ne puissent être, en tout état de cause, modifiées depuis le réseau 60, et à ce que leur modification ne puisse être effectuée que par un opérateur intervenant directement sur l'ordinateur serveur 1.

De même, les patrons d'objets XML I/O Format 61, en fonction des données effectivement utilisées par un utilisateur sur le réseau 60, supprimeront dans les données d'un objet 6 transmises sur le réseau 60 les données qui ne seraient pas utilisées par ledit utilisateur. Compte-tenu de ce que, dans la pratique, les données transmises pour un objet 6 ne représentent habituellement qu'une faible partie de l'ensemble des données de cet objet, ce procédé permet une réduction substantielle du volume des données transmises par l'ordinateur serveur 1, ce qui est particulièrement intéressant dans le cas d'ordinateur serveur 1 très chargé, tels que ceux qui se rencontrent dans l'Internet.

En outre, ce procédé permet de prendre en compte différents niveaux de sécurité attachés aux utilisateurs se connectant via le réseau 60, en associant à chaque niveau de sécurité un patron éliminant des données transmises à l'utilisateur les données non autorisées pour le niveau de sécurité de l'utilisateur.

En outre, du fait que ce procédé garantit que seules les données conformes au patron d'objet I/O Format 61 utilisé seront transmises, il n'y aura pas lieu de demander la restitution par la base de données 2, des données éliminées par le patron d'objet I/O Format 61 utilisé, et ainsi les données apparaissant dans les requêtes SQL "SELECT" effectuées par le moteur COS Engine pourront être limitées aux seules données transmises, ce qui diminuera de façon toute aussi sensible la charge d'entrées-sorties dans la base de données 2 dans l'ordinateur serveur 1.

Ce procédé permet de focaliser la requête sur les tables de la base de données 2 et les données dans ces tables 3 et 7 qui sont effectivement utilisées dans la requête d'un utilisateur. Ainsi, dans le cas où aucune donnée de la table annexe 7 n'est utilisée dans la requête, le procédé n'accèdera pas à cette table 7 pour la requête, ce qui, à nouveau, permettra d'améliorer sensiblement les performances de l'ordinateur serveur 1 mettant en œuvre le procédé de l'invention.

Ainsi, l'utilisation du format XML et d'un patron d'objet au format XML permet d'imposer des règles de sécurité sur les données entrantes, de limiter

l'utilisation de la base passante du réseau 60 aux seules données nécessaires et de diminuer en due proportion le volume des données échangées avec la base de données 2.

Dans le mode de réalisation décrit ci-dessus, la donnée "cle" 11, identifiant de façon unique un objet 6, a été indiquée, pour des raisons de simplicité de l'exposé, comme étant gérée de façon explicite par le procédé de l'invention. Toutefois, cette technique explicite n'est en aucune manière obligatoire, et elle peut parfaitement être remplacée par toute autre technique équivalente pouvant être offerte par le SGBDR utilisé.

De même, cette donnée "cle" 11 a été décrite comme étant constituée d'un seul nombre entier, mais elle peut tout aussi bien être constituée de plusieurs parties. Par exemple, dans le mode de réalisation préféré de l'invention, une donnée "cle" 11 unique, appelée Object IDentifier ou OID, est obtenue pour chaque objet 6 en concaténant une première partie constituée du nom logique du serveur 1 sur lequel réside la base de donnée 2, une seconde partie constituée du numéro de la table principale 3 dans la base de donnée 2, et une troisième partie constituée du numéro d'enregistrement de l'ensemble de données principal 31 dans ladite table 3.

Le nom logique du serveur 1 étant unique sur le réseau 60, la base de données 2 étant unique sur le serveur 1 où elle est stockée, le numéro de la table 3 étant unique dans la base de données et le numéro d'enregistrement de l'ensemble de données principal 31 étant unique dans la table 3, l'OID 11 ainsi obtenu pour un objet 6 sera donc unique parmi l'ensemble des OID identifiant les objets 6 accessibles via le réseau 60. En outre, ce type de donnée "cle" 11 présentera l'intérêt de permettre de localiser précisément le lieu de stockage d'un objet 6 quelconque sur la seule base de son OID 11 associé.

Par ailleurs, dans l'exemple décrit ci-dessus, il a été fait référence à une application de type annuaire, mais le procédé de l'invention est également utilisé dans des application de type conférences ou publications.

D'une façon générale, la description ci-dessus ne doit pas être comprise comme réduisant en quoi que ce soit la portée de la présente invention telle que revendiquée dans les revendications annexées.

## REVENDICATIONS

1. Procédé de stockage d'objets informationnels ou objets (6), dans une base de données relationnelle (2) stockée dans un ordinateur serveur (1), ladite base de données relationnelle (2) étant constituée de tables, chaque table étant constituée d'un tableau d'ensembles de données simples, lesdits ensembles ayant la même structure de données dans une même table, chaque donnée simple d'une table étant désignée par un identifiant unique dans ladite table, un objet (6) étant constitué d'une ou plusieurs données simples pouvant être stockées dans une table de ladite base de données (2), et/ou d'un ou plusieurs objets emboîtés dans ledit objet, ledit emboîtement pouvant être réalisé sur un nombre quelconque de niveaux pour réaliser un objet (6), un objet emboîté ou une donnée simple étant dit localement en nombre fixe s'il ou elle apparaît exactement une fois dans l'objet le ou la contenant immédiatement et étant dit localement en nombre variable dans le cas contraire, une donnée simple apparaissant à un niveau quelconque dudit objet (6) étant dite globalement en nombre fixe si elle est localement en nombre fixe et si tous les objets la contenant sont localement en nombre fixe, une donnée simple apparaissant à un niveau quelconque dudit objet (6) étant dite globalement en nombre variable si elle est localement en nombre variable ou si l'un quelconque des objets la contenant est localement en nombre variable, lesdites données globalement en nombre fixe d'un objet à stocker (6) étant stockées dans un ensemble de données principal (31) stocké dans une table principale (3) de ladite base de données (2), lesdites données simples globalement en nombre variable dudit objet à stocker (6) étant stockées dans une ou une ou plusieurs tables annexes (4, 5, 7) de ladite base de données (2), caractérisé par le fait que, lorsqu'elles existent, les données simples globalement en nombre variable desdits objets (6) sont stockées dans une unique table annexe (7) de ladite base de données, le procédé créant un ou plusieurs ensembles de données annexes (71) pour stocker lesdites données simples globalement en nombre variable dans ladite unique table annexe (7).

2. Procédé selon la revendication 1, dans lequel chacun desdits un ou plusieurs ensembles annexes (71) éventuellement stockés dans ladite unique table annexe (7) comporte en outre un ensemble (72) d'indicateurs booléens, chaque indicateur booléen étant associé à une donnée particulière desdites données simples globalement en nombre variable stockées dans ledit

ensemble annexe (71), ledit indicateur booléen indiquant si ladite donnée simple globalement en nombre variable associée est définie ou non dans ledit ensemble annexe (71).

3. Procédé selon la revendication 2, dans lequel certains desdits indicateurs booléens dudit ensemble (72) d'indicateurs booléens sont communs à plusieurs données simples globalement en nombre variable lorsque lesdites données simples sont localement en nombre fixe dans un même objet les contenant immédiatement.

4. Procédé selon l'une quelconque des revendications précédentes, dans lequel ledit ensemble principal (31) associé au dit objet (6) à stocker comporte une donnée (11) permettant d'identifier de façon unique ledit ensemble principal (31) dans ladite table principale (3).

5. Procédé selon la revendication 4, dans lequel ladite donnée unique (11) est unique dans ledit ordinateur serveur (1).

6. Procédé selon l'une des revendications 4 ou 5, dans lequel ladite donnée unique (11) stockée dans ledit ensemble principal (31) associé au dit objet (6) à stocker est en outre stockée dans chacun des ensembles annexes (71) associés au dit objet (6) à stocker, pour permettre la mise en relation dudit ensemble principal (3) et du ou desdits ensembles annexes (71) éventuels associés au dit objet (6) à stocker.

7. Procédé selon l'une quelconque des revendications 4 à 6, dans lequel ladite donnée unique (11) est constituée du nom logique de l'ordinateur serveur (1) sur ledit réseau (60), du numéro de table de ladite table principale (3) dans ladite base de données (2) et du numéro d'ensemble de données dudit ensemble principal (31) dans ladite table principale (3), ledit nom logique de l'ordinateur serveur (1) étant unique sur ledit réseau (60), ladite base de données (2) étant unique sur ledit ordinateur serveur (1), ledit numéro de table de ladite table principale (3) étant unique dans ladite base de données (2) et ledit numéro d'ensemble de données dudit ensemble principal (31) étant unique dans ladite table principale (3).

8. Procédé selon l'une quelconque des revendications précédentes, dans lequel deux objets (6) à stocker sont dits de même type s'ils sont constitués de données simples et d'objet emboîtés de même type, deux objets (6) de même type ayant en commun un même identifiant et les données simples et/ou les objets emboîtés se correspondant à un niveau quelconque desdits objets (6)

de même type ayant en commun un même identifiant, unique dans l'objet les contenant immédiatement.

9. Procédé selon la revendication 8, dans lequel le procédé crée un identifiant global pour tous les objets (6) de même type et pour toutes les données simples se correspondant à un niveau quelconque desdits objets (6) de même type, ledit identifiant global étant ledit identifiant commun pour lesdits objets de même type, et ledit identifiant global étant obtenu, pour chacune desdites données simples se correspondant, par la concaténation des identifiants, dans les objets les contenant immédiatement, de tous les objets contenant ladite donnée simple, et de l'identifiant de ladite donnée simple dans l'objet la contenant immédiatement.

10. Procédé selon la revendication 9, dans lequel le nombre de caractères de l'identifiant global est tronqué au nombre de caractères permis par ladite base de données (2) pour l'identifiant d'une donnée stockée dans une table de ladite base de données (2).

11. Procédé selon la revendication 10, dans lequel une ambiguïté pouvant apparaître du fait de ladite troncature est résolue en effectuant les étapes consistant à :

- remplacer le dernier caractère alphabétique de l'identifiant global par le chiffre zéro, ainsi que les éventuels chiffres le suivant ;
- augmenter d'une unité le nombre constitué par l'ensemble des chiffres apparaissant à la fin dudit identifiant global jusqu'à ce que l'ambiguïté disparaisse ou que les chiffres à la fin dudit identifiant global soient entièrement constitués de chiffres neuf ;
- répéter les étapes précédentes depuis le début lorsque les chiffres apparaissant à la fin dudit identifiant global sont entièrement constitués de chiffres neuf à l'issue de l'étape précédente.

12. Procédé selon l'un quelconques des revendications précédentes, dans lequel les objets (6) stockés dans la base de données (2) sont reçus ou transmis par l'ordinateur serveur (1) via un réseau informatique (60) de type Internet.

13. Procédé selon l'une des quelconques des revendications précédentes, dans lequel les données simples constituant lesdits objets (6) stockés dans ladite base de données (2) sont de type texte.

14. Procédé selon la revendication 13, dans lequel lesdites données simples de type texte sont des données écrites en langage XML.

15. Procédé selon la revendication 14, dans lequel lesdites données écrites en langage XML sont conformes à une description (92) de type XML Schema.

16. Procédé selon la revendication 15, dans lequel un ensemble de données (95) au format XML est obtenu par une traduction automatique de ladite description (92), ledit ensemble de données (95) et un ensemble de données complémentaire (96) étant à leur tour traduits automatiquement en langage SQL pour définir et gérer la structure de la base de données (2), et pour contrôler les échanges de données avec ladite base de données (2).

17. Procédé selon l'une quelconque des revendications 14 à 16, dans lequel une partie des données simples des objets (6) reçus par ledit réseau (60) de type Internet est supprimée à l'aide d'un patron, ou modèle, d'objet (61) écrit en langage XML, pour interdire la modification desdites données simples via ledit réseau (60).

18. Procédé selon l'une quelconque des revendications 14 à 17, dans lequel une partie des données simples des objets transmis via ledit réseau est supprimée à l'aide d'un patron d'objet (61) écrit en langage XML pour limiter la transmission sur ledit réseau (60) à certaines données simples desdits objets (6).

19. Procédé selon la revendication 18, dans lequel la suppression de d'une partie des données simples contenues dans les objets (6) permet en outre d'optimiser les requêtes à ladite base de données (2).

20. Système de stockage d'objets informationnels ou objets (6), dans une base de données relationnelle (2) stockée par un ordinateur serveur (1), caractérisé par le fait qu'il met en œuvre le procédé selon l'une quelconques des revendications précédentes.

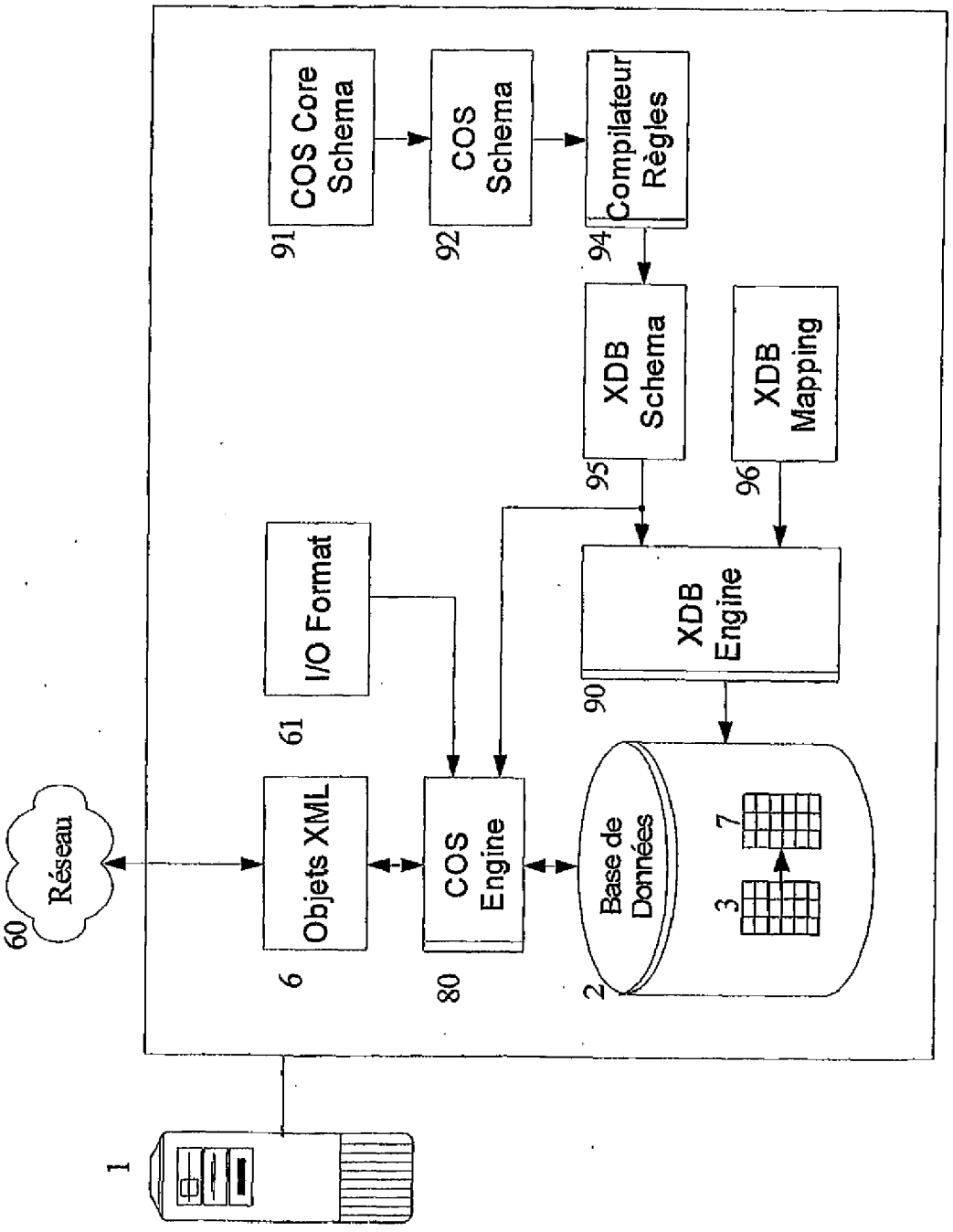

COS Core Schema
91
COS Schema
92
Compilateur Règles
94
XDB Schema
95
XDB Mapping
96
I/O Format
61
XDB Engine
90
Réseau
60
Objets XML
6
COS Engine
80
Base de Données
2
7
3
1

FIGURE 1

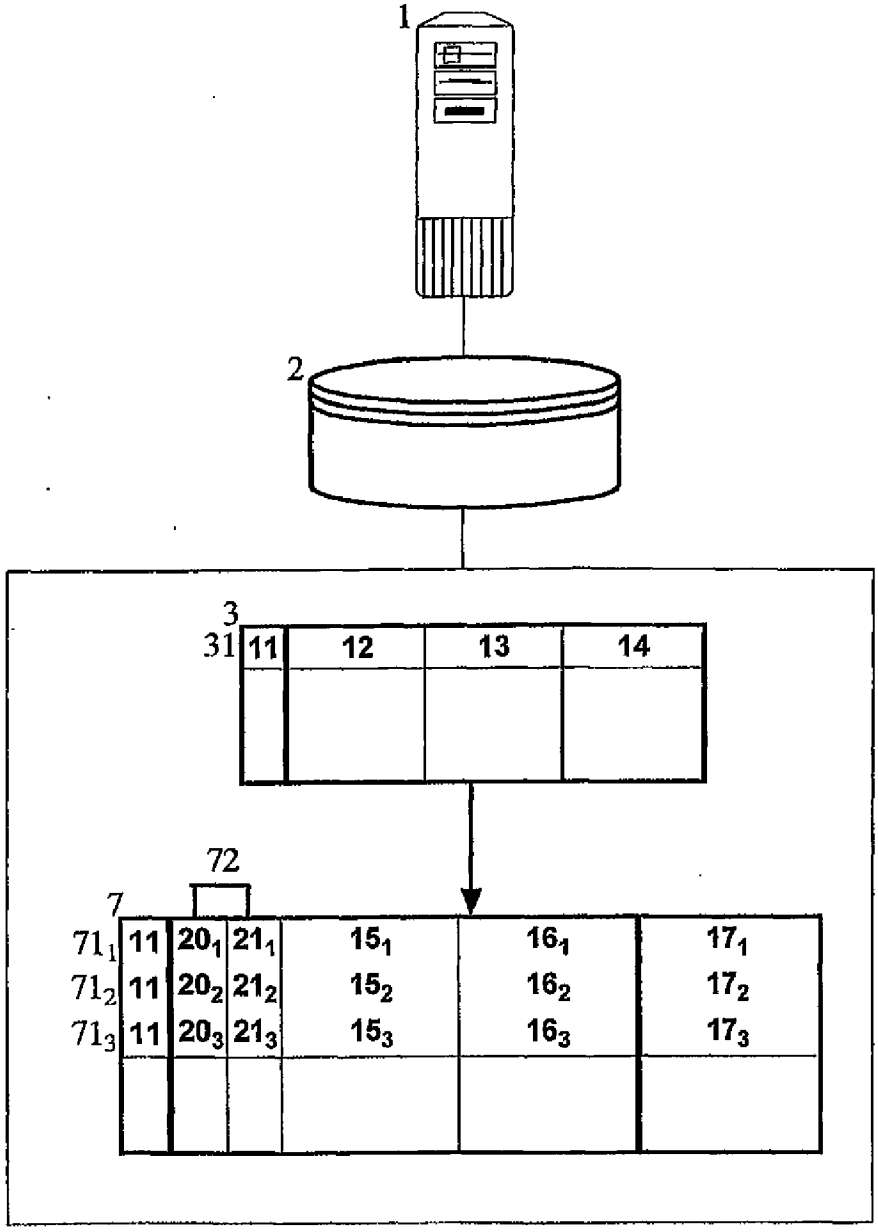

1
2
3
31
11
12
13
14
72
7
$71_1$ 11 $20_1$ $21_1$ $15_1$ $16_1$ $17_1$
$71_2$ 11 $20_2$ $21_2$ $15_2$ $16_2$ $17_2$
$71_3$ 11 $20_3$ $21_3$ $15_3$ $16_3$ $17_3$

FIGURE 2

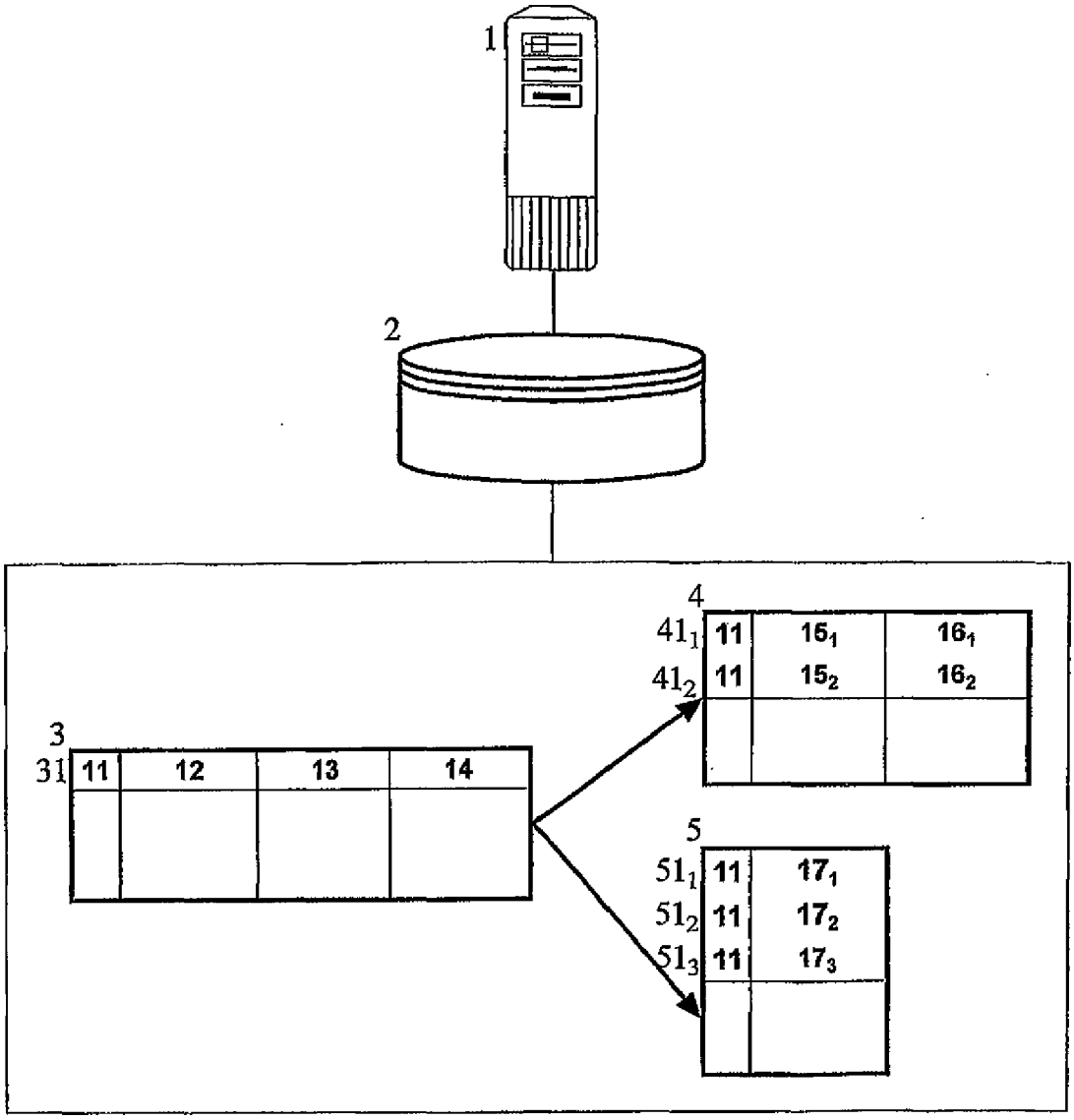

1
2
3
31
11
12
13
14
4
41₁
11
15₁
16₁
41₂
11
15₂
16₂
5
51₁
11
17₁
51₂
11
17₂
51₃
11
17₃

FIGURE 3

# INTERNATIONAL SEARCH REPORT

International Application No
PCT/FR 00/01902

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
INSPEC, EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category ° | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| A | BOURRET R ET AL: "A generic load/extract utility for data transfer between XML documents and relational databases" PROCEEDINGS SECOND INTERNATIONAL WORKSHOP ON ADVANCED ISSUES OF E-COMMERCE AND WEB-BASED INFORMATION SYSTEMS. WECWIS 2000, PROCEEDINGS SECOND INTERNATIONAL WORKSHOP ON ADVANCED ISSUES OF E-COMMERCE AND WEB-BASED INFORMATION SYSTEMS. WECWIS 2000, MILP, pages 134-143, XP002169780 2000, Los Alamitos, CA, USA, IEEE Comput. Soc, USA ISBN: 0-7695-0610-0 page 134, right-hand column, line 1 -page 135, left-hand column, line 13 page 137, left-hand column, line 47 -page 137, right-hand column, line 3 page 138, left-hand column, line 6 -page -/-- | 1-20 |

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| 15 June 2001 | 28/06/2001 |

| Name and mailing address of the ISA | Authorized officer |
|---|---|
| European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016 | Correia Martins, F |

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

| Category ° | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| | 138, right-hand column, line 56 | |
| A | US 5 983 215 A (LEI HUI ET AL)<br>9 November 1999 (1999-11-09)<br>column 1, line 53 -column 3, line 47<br>column 4, line 10-46 | 1-20 |
| A | US 5 594 898 A (DALAL KETAN ET AL)<br>14 January 1997 (1997-01-14)<br>column 2, line 26-43<br>column 7, line 11 -column 8, line 42 | 1-20 |
| A | SHIMURA T ET AL: "Storage and retrieval of XML documents using object-relational databases"<br>DATABASE AND EXPERT SYSTEMS APPLICATIONS. 10TH INTERNATIONAL CONFERENCE, DEXA'99 (LECTURE NOTES IN COMPUTER SCIENCE VOL.1677), PROCEEDINGS OF DEXA'99: 10TH INTERNATIONAL CONFERENCE AND WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, FLORENCE, I,<br>pages 206-217, XP001005987<br>1999, Berlin, Germany, Springer-Verlag, Germany<br>ISBN: 3-540-66448-3<br>page 207, line 1 -page 208, line 11<br>page 212, line 10 -page 214, line 5 | 1-20 |
| A | GUIDO N: "XML data processing and relational database systems"<br>XML EUROPE '99 CONFERENCE PROCEEDINGS, PROCEEDINGS OF XML EUROPE '99, GRANADA, SPAIN, 26-30 APRIL 1999,<br>pages 713-719, XP001006043<br>1999, Alexandria, VA, USA, Graphic Commun. Assoc, USA<br>page 714, line 12 -page 716, line 5 | 1-20 |

# INTERNATIONAL SEARCH REPORT

Information on patent family members

ational Application No
PCT/FR 00/01902

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| US 5983215 A | 09-11-1999 | WO 9850867 A | 12-11-1998 |
| US 5594898 A | 14-01-1997 | NONE | |

# RAPPORT DE RECHERCHE INTERNATIONALE

de internationale No
PCT/FR 00/01902

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
CIB 7 G06F17/30

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
CIB 7 G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si réalisable, termes de recherche utilisés)
INSPEC, EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERES COMME PERTINENTS

| Catégorie ° | Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents | no. des revendications visées |
|---|---|---|
| A | BOURRET R ET AL: "A generic load/extract utility for data transfer between XML documents and relational databases" PROCEEDINGS SECOND INTERNATIONAL WORKSHOP ON ADVANCED ISSUES OF E-COMMERCE AND WEB-BASED INFORMATION SYSTEMS. WECWIS 2000, PROCEEDINGS SECOND INTERNATIONAL WORKSHOP ON ADVANCED ISSUES OF E-COMMERCE AND WEB-BASED INFORMATION SYSTEMS. WECWIS 2000, MILP, pages 134-143, XP002169780 2000, Los Alamitos, CA, USA, IEEE Comput. Soc, USA ISBN: 0-7695-0610-0 page 134, colonne de droite, ligne 1 -page 135, colonne de gauche, ligne 13 page 137, colonne de gauche, ligne 47 -page 137, colonne de droite, ligne 3 page 138, colonne de gauche, ligne 6 -page -/-- | 1-20 |

[X] Voir la suite du cadre C pour la fin de la liste des documents

[X] Les documents de familles de brevets sont indiqués en annexe

° Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

| Date à laquelle la recherche internationale a été effectivement achevée | Date d'expédition du présent rapport de recherche internationale |
|---|---|
| 15 juin 2001 | 28/06/2001 |

| Nom et adresse postale de l'administration chargée de la recherche internationale | Fonctionnaire autorisé |
|---|---|
| Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016 | Correia Martins, F |

Formulaire PCT/ISA/210 (deuxième feuille) (juillet 1992)

C.(suite) DOCUMENTS CONSIDERES COMME PERTINENTS

| Catégorie ° | Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents | no. des revendications visées |
|---|---|---|
| | 138, colonne de droite, ligne 56 | |
| A | US 5 983 215 A (LEI HUI ET AL)<br>9 novembre 1999 (1999-11-09)<br>colonne 1, ligne 53 -colonne 3, ligne 47<br>colonne 4, ligne 10-46 | 1-20 |
| A | US 5 594 898 A (DALAL KETAN ET AL)<br>14 janvier 1997 (1997-01-14)<br>colonne 2, ligne 26-43<br>colonne 7, ligne 11 -colonne 8, ligne 42 | 1-20 |
| A | SHIMURA T ET AL: "Storage and retrieval of XML documents using object-relational databases"<br>DATABASE AND EXPERT SYSTEMS APPLICATIONS. 10TH INTERNATIONAL CONFERENCE, DEXA'99 (LECTURE NOTES IN COMPUTER SCIENCE VOL.1677), PROCEEDINGS OF DEXA'99: 10TH INTERNATIONAL CONFERENCE AND WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, FLORENCE, I,<br>pages 206-217, XP001005987<br>1999, Berlin, Germany, Springer-Verlag, Germany<br>ISBN: 3-540-66448-3<br>page 207, ligne 1 -page 208, ligne 11<br>page 212, ligne 10 -page 214, ligne 5 | 1-20 |
| A | GUIDO N: "XML data processing and relational database systems"<br>XML EUROPE '99 CONFERENCE PROCEEDINGS, PROCEEDINGS OF XML EUROPE '99, GRANADA, SPAIN, 26-30 APRIL 1999,<br>pages 713-719, XP001006043<br>1999, Alexandria, VA, USA, Graphic Commun. Assoc, USA<br>page 714, ligne 12 -page 716, ligne 5 | 1-20 |

# RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relat[illegible]x membres de familles de brevets

[illegible]de internationale No
[illegible]/FR 00/01902

| Document brevet cité au rapport de recherche | Date de publication | Membre(s) de la famille de brevet(s) | Date de publication |
|---|---|---|---|
| US 5983215 A | 09-11-1999 | WO 9850867 A | 12-11-1998 |
| US 5594898 A | 14-01-1997 | AUCUN | |